



PureConnect®

2023 R3

Generated:

09-November-2023

Content last updated:

25-October-2023

See **Change Log** for summary of changes.



Interaction Designer

Printed Help

Abstract

This document contains the application help for Interaction Designer.

For the latest version of this document, see the PureConnect Documentation Library at: <http://help.genesys.com/pureconnect>.

For copyright and trademark information, see https://help.genesys.com/pureconnect/desktop/copyright_and_trademark_information.htm.

Table of Contents

Table of Contents	2
Overview of Building and Modifying Handlers and Subroutines	22
Overview of Steps	22
Step One: Research	22
Step Two: Planning	22
Step Three: Building	23
Step Four: Activating	23
Step Five: Testing	23
Step Six: Promoting	23
Tools	24
ACD	24
Introduction to ACD Tools	25
ACD Agent Category	26
ACD Initiate Bullseye Processing	26
ACD Initiate Processing	27
ACD Interaction Details	30
ACD Reassign Interaction	31
ACD Remove Interaction Skill	32
ACD Select Agent	33
ACD Select Interaction	34
ACD Specify Interaction Category	35
ACD Specify Interaction Skill	35
ACD Statistics (Interaction)	36
ACD Statistics (Queue)	38
Agent Log On	40
Agent Log On Remote	41
Agent Log Off	42
Release Agent Lock	42
Release Interaction Lock	43
Replace Interaction	44
Reset In Queue Timeout	45
Stop ACD Processing	45
Workgroup Agent Activate	46
Workgroup Agent Deactivate	46
Accumulators	47
Accumulators	47
Get Acc. (Boolean)	48
Get Acc. (DateTime)	49
Get Acc. (Integer)	50
Get Acc. (Numeric)	51
Get Acc. (String)	52
Inc. Acc. (Integer)	53
Inc. Acc. (Numeric)	54
Lock Accumulator	55
Set Acc. (Boolean)	56
Set Acc. (Datetime)	57
Set Acc. (Integer)	58
Set Acc. (Numeric)	59
Set Acc. (String)	60
Unlock Accumulator	61
Basic	61
Introduction to Basic Tools	61
Assignment Tool	62
Condition Tool	63
Notify Debugger	64
Selection Tool	65
Table Lookup	65
Write Trace Message	68
Buffer	68
Introduction to Buffer Tools	69
BufferGetDate	69
BufferGetInteger	71
BufferGetLength	72

BufferGetString	72
BufferHandleToInteger	74
BufferPutDate	75
BufferPutInteger	76
BufferPutString	77
CopyBuffer	78
CreateBuffer	78
DecodeBuffer	79
DeleteBuffer	79
EncodeBuffer	80
IntegerToBufferHandle	80
Calendar	81
Introduction to Calendar Tools	81
Add Event	82
Add Task	83
Are Available	84
Delete Event	85
Delete Task	85
Edit Event	86
Edit Task	87
Find Next	88
Get Event	88
Get Events	90
Get Task	91
Get Tasks	92
Is Available	93
Logon	94
Open Calendar	95
Search Events	96
Search Events Ex	96
Search Tasks	98
Search Tasks Ex	99
Database	99
Database Tools	99
The Order in Which Database Tools Should Be Used	101
DB Stored Procedure Definitions	101
Supported Databases	102
Sample SQLExec Statements	103
Sample Where Clauses	103
Database Tools	103
Dialer	120
Dialer Call Completed	120
Get Campaigns	123
Get Dialer Attributes	124
Get Logged In Agents	125
Place Dialer Call	125
Restart Campaign	128
Set Dialer Attributes	129
Director	129
Director Tools and Initiators in Interaction Designer	130
Director Monitored Server Ready tool	131
Director Select Queue Tool	131
Play Audio File (No Conference) tool	133
Send ATT Announcement Code Tool	134
Send ATT Post Feature Code Tool	135
Send MCI DDD Tool	136
Send MCI Destination w/DNIS Override Tool	137
Send MCI Error Tool	138
Send MCI IDDD Tool	139
Send MCI Use Default Tool	140
Send Route Command Tool	141
Send SIP Destination	142
Send Simulated Route Command Tool	143
Send Sprint Error Tool	144
Send Sprint Reject Tool	145
Send Sprint Use Default Tool	146

Director Initiators	146
Internal Use Only Tools	158
Email	162
Introduction to E-mail Tools	162
Change Message Status	164
Change Message Status By Cookie	165
Create Folder	166
Delete Folder	167
Delete Message	168
Delete Message By Cookie	169
Email Interaction Create	169
Email Interaction Disconnect	171
Email Interaction Get Message	172
Email Interaction Hold	173
Email Interaction Insert Attachment	174
Email Interaction Park	175
Email Interaction Record	176
Email Interaction Send Message	177
Email Interaction Transfer	178
Email Interaction Update Message	178
File To Recording	181
Find Message	181
Find Messages	183
Forward Message	184
Get Contained Folders	188
Get Cookie From Message	189
Get Message Count	189
Get Message From Cookie	191
Get Out of Office Status	192
Get Quota	193
Get Quota Resources	194
Get Quota Roots	195
Get Recipient Info	196
Is Distribution List Member	197
Mail Exchanger DNS Lookup	198
Make Attached File	199
Make Cookie	200
Make Email Body	201
Move Message	202
Move Message Ex	203
Open Attached File	204
Open Attached Message	205
Open Folder	205
Open Message	208
Open Message By Cookie	210
Parse Cookie	212
Parse Email Body	213
Query Mail System	214
Reply To Message	214
Send E-Mail	217
Send Fax	220
Send Message Light Notification	224
Send Voicemail	224
Send VPIM Message	227
Set Out of Office Status	230
Update Folder	230
Fax	232
Introduction to Fax Tools	232
Fax Tool Usage	234
Supported Bitmap File Formats	234
Interaction Fax format	235
Fax Tools	235
Feedback	252
Introduction to Feedback Tools	252
Get Active Survey	253
Get Survey TUI XML	253

IsSurveyActive	255
Manage Survey Prompt Result	256
Post Survey Results	257
Run Survey	258
Transfer Survey Queue	259
Wait for Survey Start	260
File I/O	260
Introduction to File I/O Tools	261
TCP/IP tools overview	262
Copy File	263
Directory List	264
File Attributes Modify	265
File Attributes Query	266
File Close	267
File Delete	267
File Find Replace	267
File Open Read	269
File Open Write	270
File Position	271
File Read	272
File Read List	272
File Write	274
File Write List	275
Get File Statistics	276
Get Free Space	277
TCP Close	277
TCP Connect	278
TCP Listen	278
TCP Read Integer	279
TCP Read String	279
TCP Read String UTF8	280
TCP Write Integer	282
TCP Write String	283
TCP Write String UTF8	284
Generic Object	284
Introduction to Generic Objects Tools	284
Create Generic Object	285
Disconnect Generic Object	286
Get Generic Object Attributes	287
Set Generic Object Attributes	288
Transfer Generic Object	288
Host Interface	289
Introduction to Host Interface tools	289
Host Connect	293
Host Connect Ex	293
Host Disconnect	295
Host Disconnect Ex	295
Host Fetch Form Data	296
Host Find String	297
Host Get Connection Resource	298
Host Get Cursor Position	299
Host Get Field Attributes	299
Host Get Resource Counts	301
Host Get Screen	302
Host Get Screen Dimensions	303
Host Get String	304
Host Initialize Resource	305
Host Move To Position	306
Host Move To Screen	307
Host Press Key	308
Host Put Form Data	309
Host Put String	310
Host Verify Screen	311
Host Wait For Cursor	312
Host Wait For ReadyToSend	313
Host Wait for Screen	314

Host Wait For String	315
ICon	315
Overview of Interaction Conference (ICon) Tools	316
ICon Handle Call	316
Internet	316
Internet tools	316
Create Callback	317
Escape URL	318
Generate HTML	318
UnEscape URL	320
IpNotes	320
Overview of IpNotes	320
Create Note	321
Get Notes Attribute Date/Time	322
Get Notes Attribute Integer	323
Get Notes Attribute String	324
Remove Note	325
Update Notes Attribute Date/Time	326
Update Notes Attribute Integer	327
Update Notes Attribute String	328
LDAP	329
Introduction to LDAP tools	329
LDAP: The order in which LDAP tools might be used	333
Add Blob Attribute	334
Add Entry	335
Add Entry Ex	336
Add Operation	336
Close Session	338
Delete Entry	339
Explode DN	340
Flush Cache	341
Get Cache Size	342
Get Entry Attribute	343
Get Entry Blob Attribute	344
Login	344
Modify Entry	346
Modify Entry Ex	347
Move Entry	347
Next Entry	349
Read Entry	351
Rename Entry	352
Search Entries	352
Session	354
Set Max Cache Size	356
List	356
List Tools	356
Copy String List	358
Filter List	358
Find	362
GetAt	363
GetCount	363
InsertAt	364
InsertAtHead	364
InsertAtTail	365
List to String	365
Merge String Lists	366
Parse String	367
RemoveAll	367
RemoveAt	368
RemoveHead	368
RemoveTail	368
SetAt	369
Sort Lists	370
Update Data Pair Values	371
Outputs	371
Monitoring (Remoco)	371

Get Process Information	371
Merge Log Event ID and Insertion Strings	373
Merge Log Event Messages	374
Restart IC Subsystem	374
Stop IC Subsystem	375
Multi-Site	377
Multi-Site Create Message	377
Multi-Site Get Integer	377
Multi-Site Get Note	378
Multi-Site Get String	379
Multi-Site Put Integer	379
Multi-Site Put Note	380
Multi-Site Put String	380
Multi-Site Send Event	381
Multi-Site Send Request	382
Multi-Site Send Response	383
OCR	383
Overview of OCR Tools	383
Export for OCR file	384
Export Parser	386
OCR for I3Fax files	387
OCR for TIFF/PCX/DCX files	389
OCR Parser	391
Personal Rules	394
Overview of Personal Rules	394
Personal Action Details	394
Personal Rules	395
Process Automation	395
Overview of Process Automation Tools	396
Create Data Container	397
Get Data Element	397
Get Process Properties	400
Initiate Process	400
Send Process Automation Handler Results	403
Put Data Element	404
Query Server Info	404
Remove Data Container	406
Reco	406
Introduction to Reco tools	407
Reco Add Preloaded Grammar	407
Reco Analyze Error Code	409
Reco Analyze Result	409
Reco Basic Input	410
Reco Bind Slot Values	414
Reco Create Company Directory Grammar	414
Reco Create Company Directory Grammar 2	417
Reco Create Simulated Interaction	419
Reco Custom Operation	421
Reco Disconnect Simulated Interaction	422
Reco Filter Result	422
Reco Get Hypothesis	423
Reco Get Next Hypothesis	425
Reco Get Property	426
Reco Get Registered Grammars	427
Reco Get Slot Value	427
Reco Has Feature	429
Reco Initialize	429
Reco Input	432
Reco Log Event	436
Reco Merge Results	436
Reco Query Input Modes	439
Reco Query Simulator Script State	440
Reco Register Grammar	440
Reco Register Grammar String	442
Reco Register Inline Grammar	444
Reco Register Preloaded Grammar	446

Reco Release	448
Reco Session Active	448
Reco Set Inline Properties	449
Reco Set Input Modes	449
Reco Set Property	451
Reco Set Simulator Script State	452
Reco Unregister All Grammars	453
Reco Unregister Grammar	454
Reco Verifier Abort Training	455
Reco Verifier Commit Training	456
Reco Verifier Ignore Last Utterance	457
Reco Wait For Grammars	457
Reports	458
Report Tools	459
Log Tools	460
Advance Counter	461
Assign Report Group	461
Assign Stats Group	462
Get Nth Period Statistics Report Data	462
IVR Event Notify	462
IVR_EnterLevel	464
IVR_ExitIVR	465
Logging Custom Passthrough	466
Purge Log Records	467
Query Counter	467
Query Report Group	468
Query Stats Groups	468
Remove Stats Group	469
Report E-mail	469
Report Export File	472
Report HTML Export	474
Report Print	475
REST	477
Array Builder	478
Array Parser	479
Bearer Token Request	479
JSON Builder	482
JSON Parser	483
REST HTTP Request	483
SMS	485
Overview of SMS Tools	486
SMS Append Destination	486
SMS Append Destinations	487
SMS Append Message	488
SMS Complete Send	489
SMS Create	489
SMS Get Results	490
SMS Send	490
SMS Set Message	493
Inputs	493
SOAP	493
SOAP Abort Request	494
SOAP Add Body Element	494
SOAP Add Header Element	495
SOAP Add RPC Parameter	497
SOAP Base64 Decode	498
SOAP Base64 Decode To File	499
SOAP Base64 Encode	500
SOAP Base64 Encode File	501
SOAP Create Array	501
SOAP Create Envelope	504
SOAP Create Fault Response	505
SOAP Create RPC Response	506
SOAP Expects Response	508
SOAP Get Body	508
SOAP Get Body Element	508

SOAP Get Fault	510
SOAP Get Header	511
SOAP Get Header Element	511
SOAP Get Header Elements	512
SOAP Get Next RPC Parameter	514
SOAP Get Request Info	515
SOAP Get RPC Method Info	516
SOAP Get RPC Parameter	517
SOAP Get Transport Info	518
SOAP HTTP Request	518
SOAP HTTP Request Ex	521
SOAP HTTP Request Ex2	524
SOAP HTTP Request Ex3	528
SOAP Parse Request Payload	531
SOAP Query Encoding Style	534
SOAP Send Response	535
SOAP Set Element Type	536
SOAP Set Fault	537
Schedules	537
Get Schedules	537
Get Best Schedule	539
StatAlertServer	540
Alert Custom Handler Result	540
System	540
System Tools	541
Complete External Blind Transfer	541
Complete External Call and Complete External Call (extended)	544
Complete External Call (extended)	545
Complete Intercom Blind Transfer	546
Complete Intercom Call	547
DataManager Query	547
Delete DS Key	549
Dialplan Failure	549
DID/DNIS Routing	550
DLL Function Call (Out of process)	550
Execute Shell Command	551
External Handler Return	553
Feature Licensed?	553
GetDSAttr	553
GetDSAttrs	555
Get Ds Keys	556
Get DS Parameter	557
Get Email Profile	558
Get Operator Profile	559
Get Profile	559
Get Structured Parameter	561
Keypad Map	562
Load Localized String	563
Log Event	564
LookUp	564
LookupList	567
Lookup List Extended	568
Make External Handler Call	570
OCR Installed	572
Parse String RegEx	573
Put Ds Attr	574
Put Ds Attrs	575
Put Ds Key	575
Put External Password	578
Put Structured Parameter	579
Query Backup	580
Query Conversation ID First Agent	581
Query ConversationID Last Agent	582
Query License Details	583
Query Security Policy	583
Quick Directory Available Check	585

RegEx Extended	585
Retrieve External Password	588
Semaphore Lock	589
Semaphore Unlock	590
Send Custom Notification	591
Server Name	591
Sleep	592
Tracker User Audit Data	592
Tracker VoiceMail Data	592
Transcribe Recording Begin	593
Transcribe Recording Result	595
Unique ID	597
Verisign Transaction	597
Whitepages	597
WhitePagesAsync	602
WhitePagesLocality	604
TUI	604
Attendant Email Logical Transfer	604
Create Calendar Session	605
Create Private Contact Grammar	605
Find Localized File	608
Find User By TN	609
Get File Age	609
TUI Compile from String	610
TUI Get Catch	610
TUI Get Menu Attributes	610
TUI Session	610
Telephony	610
Telephony	610
Add Party	613
Alert	613
Alert Station Group	617
Alert Workgroup	619
Answer	622
Assemble Prompt Phrase	622
Assemble String from Attributes	625
Assemble Text Phrase	626
Auto Conference	627
Bind Provisional Station	628
Blind Transfer	629
Change Workgroup Queue Initiator	630
Exit Paths	630
Complete Parallel Make Call	630
Compress Audio File	630
Conference	632
Convert Call Id to String	633
Convert Conference ID to String	633
Convert String to Call Id	634
Convert String to Conference ID	635
Deferred Answer	636
Describe Phone Number	636
DID/DNIS Routing Ex	639
Disconnect	640
Extended Blind Transfer	640
Extended Get Key	643
Extended Get Key Async	645
Extended Place Call	646
Flush Audio	650
Flush Keys	651
Get Attribute	651
Get Attributes	652
Get Billing Rate	653
Get Call Log	653
Get Datetime Attribute	654
Get Datetime Attributes	654
Get Key	654

Get Station Info	656
Get User's Location	657
Get Users with Role	658
Get Wildcard Attributes	659
Hold	660
Is Alertable?	660
Key Word Spotting	663
Listen From Call	664
Listen From Station	665
Log Message	666
Malicious Call Trace	667
Mute Interaction	668
Parallel Make Call	668
Parallel Make Call Failure	671
Park	671
PickUp	672
Place Call	673
Play Audio File	673
Play Digits	675
Play Prompt	676
Play Prompt Extended	677
Play Prompt Phrase	677
Play Recording	679
Play String	680
Play String Extended	680
Play Text File	682
Play Text File Extended	682
Play Tone	684
Priority Set Attribute	684
Priority Set Attributes	684
Private	685
Query Conference Properties	686
Query Logged In Users	687
Query Media Subtype	688
Query Media Type	688
Query Monitored Queues	690
Query Queue	690
Query Queue Type	692
Query Statuses for User	693
Query User Status	693
Record Audio	694
Record Call	697
Record File	698
Record String	700
Record String Extended	700
Record Text File	702
Record Text File Extended	702
Reload Station	704
Remove Party	704
Reset Password	705
Secure Session Begin	706
Secure Session End	707
Secure Session Get Key	707
Secure Session Info Insert	710
Inputs	710
Secure Session Info Validate	711
Select Call	712
Send ADSI String	713
Set Attribute	714
Set Attributes	715
Set Billing Rate	716
Set Call State	717
Send ADSI String	718
Set Datetime Attribute	719
Set Datetime Attributes	719
Set DTMF Password	719

Set State String	721
Set User Status	721
Set Visual Indicator	723
Start TDD	723
Station Answer	724
Station Audio	725
Station Connection Confirmation	726
Station Group Pickup	726
Station Pickup	727
Station Place Call	728
Synchronous Answer	729
System Queue	729
Transfer	730
Unbind Station	731
Unhold Call	731
User Login List	732
Validate DTMF Password	733
Verify Conference ID	734
Verify Interaction ID	734
Wait For Call On Queue	735
Wait For Disconnect	736
Wait For Monitor End	737
Wait Wrap Up	738
Wink	738
Zone Page	738
UMF	740
Overview of Universal Messaging Facility	740
UMF Create Message	741
UMF Get Integer	741
UMF Get String	742
UMF Put Integer	742
UMF Put String	743
UMF Send Event	744
UMF Send Request	745
UMF Send Response	746
VoiceXML	746
VoiceXML Tools Overview	747
VoiceXML Async Initiate	747
VoiceXML Async Initiate Document	749
VoiceXML Initiate	750
VoiceXML Initiate Document	752
VoiceXML Async Cancel	755
Web Interaction Tools	755
Overview of Web Interaction Tools	756
Alert Interaction	757
Chat Goto URL	758
Chat Send File	759
Conference Interaction	759
Consult Transfer	760
Create Interaction	761
Disconnect Interaction	762
Hold Interaction	762
Inputs	762
Mute Interaction	763
Park interaction	763
Pickup Interaction	764
Receive Text	764
Receive Text Async	765
Record	766
Send Text	767
Send To Voicemail	767
Snip Recording	768
Snooze Interaction	769
Transfer Interaction	770
WebSphere MQ	770
Overview of WebSphere MQ Tools	771

MQ Begin	772
MQ Close	773
MQ Commit	774
MQ Connect	774
MQ Disconnect	776
MQ Extended Get	776
MQ Extended Put	779
MQ Flush	782
MQ Get	782
MQ Open	784
MQ Put	785
MQ Rollback	787
XML	787
XML Add Schema	788
XML Assign Node	789
XML Assign Node Iterator	789
XML Clone Node	790
XML Create Document	790
XML Create Document From String	792
XML Create Node	793
XML Escape Entities	795
XML Get Attribute	797
XML Get Attributes	798
XML Get Child Nodes	799
XML Get Document Property	800
XML Get Error Info	800
XML Get First Iterator Position	802
XML Get Namespaces	803
XML Get Next Node	804
XML Get Node Info	804
XML Get Node Value	806
XML Get Schema	807
XML Get Sibling	808
XML Get Text	809
XML Get XML	810
XML Insert Node	811
XML Load Document	811
XML Remove Node	813
XML Save Document	814
XML Select Nodes	815
XML Select Nodes As List	816
XML Select Single Node	817
XML Select Single Node Set Value	818
XML Set Attribute	818
XML Set Document Property	819
XML Set Node Value	821
XML Switch On Node Type	822
XML Transform To Document	822
XML Transform To String	825
XML Unescape Entities	826
XML Validate Document	827
Initiators	827
Introduction to Initiators	827
ACD Agent Available Initiator	831
ACD Call Timeout Initiator	832
ACD Process Call Initiator	833
Call Monitor Initiator	833
Call to Non-System Queue Initiator	836
Client Button Press Initiator	838
Client Prompt Request initiator	839
Compile Voicemail TUI Initiator	840
Complete Async Receive Text from an Interaction Initiator	841
Confirm Station Connection	842
Continuous Monitor Request	843
Custom Notification Initiator	844
Directory Services Change Notification Monitor Initiator	844

Email Interaction Disconnected Initiator	847
Email Interaction Incoming Initiator	848
Email Interaction Outgoing	849
Email Interaction Transferred Initiator	850
External Handler Call Initiator	851
Fax Send Completed	851
Generic Object In Non System Queue Initiator	855
Generic Object Monitor Initiator	856
Get Digits Ex Async Initiator	856
Held Interaction Timer Initiator	858
HTML Event Initiator	858
IC Change Notification Monitor Initiator	861
Incoming Call Initiator	863
Incoming Fax Initiator	863
Incoming Mail Initiator	865
Incoming SMS	865
Incoming Status Report	868
Incoming VPIM	869
Interaction Administrator Change Notification Monitor Initiator	869
Interaction Disconnected Initiator	871
Interaction Retry Later Attempt initiator	872
Interaction Snooze Timed Out initiator	873
Interaction Transferred to Queue Initiator	874
Keyword Spotted Initiator	874
Manage Survey Prompt Initiator	877
Message Light Notification initiator	878
Messaging Request Initiator	879
Multi-Site Message Received	880
New Incoming Interaction Initiator	881
New Incoming Web Session Initiator	882
Object Disconnect Initiator	883
Outgoing Call Request Initiator	883
Outgoing Fax Initiator	884
Outgoing SMS	887
Parallel Make Call Outbound Call Initiator	887
Play Station Audio Request initiator	889
Process Automation Initiator	890
Provision Station Initiator	891
Queue Period Report Statistics Initiator	892
Receive Log Events Initiator	892
Receive Trap Initiator	894
Run Survey Initiator	895
Secure Input Initiator	896
Send to Voice Mail Initiator	897
SMS in Non System Queue Initiator	898
SMS Monitor Initiator	899
SOAP Request Initiator	899
StatAlertServer Initiator	899
Station Off Hook initiator	902
Subroutine Initiator	903
Switchhook Flash Initiator	904
Switchover Event Initiator	905
System Initialization initiator	905
T1/E1 Wink Event Initiator	906
TCP/IP Connection Accepted initiator	907
Timer Initiator	908
Transfer Request Initiator	908
Transfer to System Queue Initiator	910
Transfer Conversation Initiator	911
Transfer Direct Message Initiator	912
UMF Message Received Initiator	913
User Status Monitor Initiator	913
Work Item Transferred to Queue Initiator	916
Wrapup Required Initiator	916
System_IncomingSocialMediaConversation	917
System_IncomingSocialMediaDirectMessage	918

Procedures	919
Starting Interaction Designer with command line parameters	919
Working with Handlers and Subroutines	919
Overview of Building and Modifying Handlers and Subroutines	919
Introduction to Handlers	921
Create a new handler or subroutine	923
Handler Standards	923
Save a handler or subroutine	926
Publish a handler or subroutine	926
Publishing a handler created in a previous release of CIC	928
Intermediate Publish	928
Using .i3pub Files	930
EICPublisher	931
Batch Publishing	932
Activating and Deactivating Handlers	934
View Dependencies	936
Views Preferences	937
Customizing Handlers	937
Retrieving the values of server and system parameters from handlers	942
The difference between server and system parameters	942
Debugging Handlers	955
Working with Subroutines	959
Managing Handlers	961
Handler Best Practices	964
Working with Steps	973
Introduction to steps	974
Add a step	975
Delete a step	975
Connect a step	975
Move a step	976
Edit a step	976
View the properties of a step	976
Find a step	977
Copy a step	977
Cut a step	977
Paste a step	978
Delete a link	978
Working with Audio Prompts	978
Prompt Editor	978
Record a prompt	980
Listen to a prompt	980
Delete a prompt	981
Importing and Exporting Prompts and Strings	981
Localizing Prompts	983
Prompt Libraries	984
Language Codes	984
Other reference	985
Retrieving the values of server and system parameters from handlers	985
The difference between server and system parameters	985
Call Analysis	985
Where is call analysis turned on and off?	986
What does call analysis actually do?	986
What is answering machine detection?	986
Is call analysis different on digital lines?	987
What happens if I turn Call Analysis Off?	987
Object States	987
States	988
Tools impacted by the PureConnect data privacy feature	988
IP Tool Tracing Changes	989
Interaction Designer interface	991
Introduction to the Design Palette	991
The Tools Page	991
The Subroutines Page	991
Disabled Features in Interaction Designer	992
Features not available without a server connection	992

File Menu Commands	992
Edit Menu Commands	993
View Menu Commands	994
Layout Menu Commands	994
Utilities Menu Commands	995
Debug Menu Commands	996
Window Menu Commands	996
Help Menu Commands	997
Toolbars	997
Designer Preferences Dialog Box	997
Expressions, Data Types, and Operators	997
Creating Expressions	997
Literal Values, Variables, and Operators	999
Expression Editor Assistant	999
Data types	1000
Miscellaneous topics	1021
Print Preview toolbar	1021
Print	1021
Next Page	1021
Prev Page	1021
One Page / Two Page	1021
Zoom In	1021
Zoom Out	1021
Close	1021
Status Bar	1022
Compile Voicemail TUI Initiator	1022
Email Tool Result Codes	1022
Align Bottom	1024
Align Left	1024
Align Right	1024
Align Top	1024
Exit command (File menu)	1024
Shortcuts	1024
Associate Call	1024
Inputs	1024
Outputs	1025
Exit Paths	1025
Auto Label Power Tools	1026
BMP file format for faxes	1026
Call ID to Integer	1026
Call Info Request	1026
Callback	1026
Calls to Subroutines	1027
Inputs	1027
Outputs	1027
Exit Paths	1027
Call Stack	1027
Choosing an Operator	1028
Typing values for operands	1028
Conference ID to Integer	1028
Context Help command	1028
Debug Palette	1029
Debug Preferences	1029
Initial Breakpoint	1029
When Stopping a Debug Session	1029
Watch Variables	1029
Debug Toolbar	1030
Call Wrap Up	1031
Initiator Properties Page	1031
Outputs	1031
Exit Paths	1031
Fax Message Initiator	1031
Initiator Properties Page	1031
Outputs	1032

Exit Paths	1032
Get Logged In Workflow	1033
Inputs	1033
Outputs	1033
Exit Paths	1033
Get Workflows	1034
Outputs	1034
Exit Paths	1034
Rule Action Event	1034
Initiator Properties Page	1034
Outputs	1034
Exit Paths	1035
Disconnect Email Object	1036
External Document Not Found	1036
Drag a tool from the tool palette to create a step	1036
Introduction to E-mail Objects	1036
End TDD	1036
EPS file formats for faxing	1036
Exporting Dependencies to XML	1036
Change Initiator command (File menu)	1037
Close command (File menu)	1037
File Debug Immediate	1037
File Download A Handler From Server Command	1037
File Generate i3pub File	1037
1, 2, 3, 4 command (File menu)	1037
New command (File menu)	1037
Open command (File menu)	1037
Print Preview command (File menu)	1038
Print Setup command (File menu)	1038
File Print command	1038
File Properties	1038
Save As command (File menu)	1038
Save command (File menu)	1038
File Export To XML	1038
Find	1038
Get Best Site Queue	1039
Inputs	1039
Outputs	1039
Exit Paths	1039
Get Campaign UUID	1040
Inputs	1040
Outputs	1040
Exit Paths	1040
Get Key Word Set Guid List	1041
Inputs	1041
Outputs	1041
Exit Paths	1041
Get Site Queue Expected Wait	1041
Inputs	1041
Outputs	1042
Exit Paths	1042
Get Site Queue Info	1043
Inputs	1043
Outputs	1043
Exit Paths	1045
Get Site Queue State	1046
Inputs	1046
Outputs	1046
Exit Paths	1046
Get Email Object Original Message	1046
Get Email Object Response Message	1047
Go TopLeft	1047
Go Line Down	1047
Go Line Left	1047

Go Line Right	1047
Go Line UP	1047
Go Page Down	1047
Go Page Left	1047
Go Page Right	1047
Go Page Up	1047
Grid	1047
Handler Diff Power Tool	1048
Handler Download Preferences	1049
Handler Retrieval	1049
Destination Directory	1049
Handler Variables	1049
How Paging Works in CIC	1050
Welcome to Interaction Designer	1051
What is Interaction Designer?	1051
Other Documentation	1051
Interact!	1051
Incomplete Required Parameter dialog box	1051
Insert Email Object Response Attachment	1051
Integer to Call ID	1052
Integer to Conference ID	1052
Introduction to Paging in CIC	1052
ISO 3166 Country Codes	1052
ISO 639 Language Codes	1059
JPG file format for faxes	1063
Key codes for use with the Host Press Key tool	1063
Keyword Spotting by Sets	1067
Inputs	1067
Exit Paths	1067
Layout Toolbar	1067
Layout Toolbar	1068
Listen	1069
Inputs	1069
Exit Paths	1069
Literal Values	1070
Using Literal Values in complex expressions	1070
Special Characters in Expressions	1070
MAC, IMG, and ISP file formats for faxing	1070
Import Global Variables	1070
Manual Dial Request Initiator	1071
Initiator Properties Page	1071
Outputs	1071
Exit Paths	1071
Mathematical Operators	1071
? + ?	1071
? - ?	1072
? * ?	1072
? / ?	1072
? ^ ?	1072
Abs	1072
Mod	1072
- ?	1072
Melder - Request Audio Path	1073
Initiator Properties Page	1073
Outputs	1073
Exit Paths	1073
Melder - Setup Audio Path	1074
Initiator Properties Page	1074
Outputs	1074
Exit Paths	1074
MP3 files	1074
Open an existing handler or subroutine	1074
Operator Descriptions	1075
Overview of Buffer Tools	1075

Overview of Multi-Site Tools	1076
Palette Toolbar	1076
PCD and FPX file formats for faxes	1077
PCT file formats for faxing	1077
PCX file formats for faxes and OCR	1077
PNG file formats for faxing	1077
Power Tools command (Utilities menu)	1077
Predictive Call Completed	1078
Inputs	1078
Exit Paths	1078
Preview Call	1079
Inputs	1079
Exit Paths	1079
Print a handler	1079
Print Magnification	1079
Printer Layout Options	1079
Printing a handler across multiple pages	1080
Printing a handler on a single page	1080
Properties	1080
PSD file formats for faxing	1080
Publish Command (File Menu)	1080
General Preferences	1081
Publish Handler Dialog	1081
Starting Directory for File Open and Save As	1081
DB Query Step Database Operations	1081
Publishing IC 2.2 handlers in IC 2.3	1081
Publishing IC 2.2 handlers in IC 2.4x	1082
QuickJump	1082
List all steps	1082
List Steps With No Incoming Links	1082
List Steps With Unlinked Exit Paths	1082
Clear QuickJump List	1082
RAS file formats for faxing	1082
Record Email Object	1082
Register Call Data	1083
Inputs	1083
Exit Paths	1083
Relabel Power Tools	1084
Remove Email Object Response Attachment	1084
Sample.lst	1084
Select All	1084
Send Message Extended	1084
Send Pager Message Initiator	1085
Initiator Properties Page	1085
Outputs	1085
Exit Paths	1085
Send Pager Message Lookup Initiator	1086
Initiator Properties Page	1086
Outputs	1086
Exit Paths	1086
Send Email Object Response	1087
Set MWI Extended 2	1087
Set MWI Extended	1087
SMDI : Send Message	1087
SOAP Wizard	1088
SOAP Tool Variables	1088
Web Services Invoked by SOAP	1088
ISAPI Listener Files	1088
Proxy Handler Options	1088
Special Export Options	1089
Explanation of Special Options	1089
Standard Toolbar	1091
Status Indicator	1091
Step Variables	1092

Step XML Power Tool	1092
STID	1092
Subroutine Parameter dialog box	1093
Parameter Label	1093
Variable Name	1093
Type	1093
Option: This parameter will only be used as an input value to this subroutine	1093
Default Value for Caller	1093
Default Entry	1093
Option: Constant Value	1093
Option: Variable Name	1093
, symbol	1093
/ symbol	1093
TextFormat codes for OCR Export	1094
TGA file formats for faxing	1095
The order in which TCP/IP tools should be used	1095
TIF file format for faxes and OCR	1096
SMDI: Set MWI	1096
Inputs	1096
Exit Paths	1096
Toggle Messages	1096
Tones	1097
Toolstep Info Power Tool	1097
Transfer Email Object	1099
TTS Speech Modes	1099
Update Email Object Response	1102
User Queue Statistics Monitor Initiator	1102
Initiator Properties Page	1102
Outputs	1102
Exit Paths	1104
Using MRCP Tools	1104
Non-Extended Tools	1104
Extended Tools	1104
Example	1105
Using the SOAP Proxy Wizard	1106
Debug command (Utilities menu)	1106
Manage Handlers command (Utilities menu)	1106
Utilities Stored Procedures	1106
Variable Bar (View Menu)	1106
Deleting Unused Variables In a Handler	1108
Variables	1109
Declaring Variables in a handler	1109
Global Variables	1109
Related Topics	1109
Messages Palette	1109
Handler Messages	1109
General Messages	1109
Status Bar command (View menu)	1109
Toolbar command (View menu)	1110
Watch Variables	1110
Handler Changes in CIC 4.0	1110
Updated Handlers	1110
New Handlers	1118
Deprecated Handlers	1119
WMF file formats for faxing	1121
Workgroup Queue Statistics Monitor Initiator	1121
Initiator Properties Page	1121
Outputs	1121
Exit Paths	1123
WPG files formats for faxing	1123
Zoom 10	1123
Zoom 100	1123
Zoom 125	1123
Zoom 150	1123

Zoom 175	1123
Zoom 200	1123
Zoom 25	1124
Zoom 50	1124
Zoom 75	1124
Zoom In	1124
Zoom Out	1124
Zoom	1124
Zoom Factor	1124
File Open Dialog Box	1125
File Name	1125
List Files of Type	1125
Drives	1125
Directories	1125
Network...	1125
Specifying Report Parameters	1125
Determining Which Parameters are Used in a Report	1125
Examples of Boolean Parameter Names and Values	1126
Examples of String Parameter Names and Values	1126
Examples of DateTime Parameter Names and Values	1127
Examples of Number Parameter Names and Values	1128
Substitution Fields	1128
Tool Changes in CIC 4.0	1128
ACD Tools	1128
Alert Server Tools	1128
Email Tools (new tools)	1128
Email Object Tools	1129
OCR Tools (new)	1129
Reco Tools	1129
Report Tools	1129
SDMI Tools	1129
StatAlertServer Tools (new tools)	1129
Web Interaction Tools (new tool and initiator)	1129
Initiators	1129
Using ITTS Tools	1129
Non-Extended Tools	1130
Extended Tools	1130
Change log	1132

Overview of Building and Modifying Handlers and Subroutines

This section offers an overview of the entire process of building and editing handlers and subroutines. You should read this before you begin using Interaction Designer to build or modify handlers and subroutines. This section contains the following subsections:

- Overview of Steps
- Step 1: Research
- Step 2: Design
- Step 3: Building
- Step 4: Activating
- Step 5: Testing
- Step 6: Promoting

Overview of Steps

The process of building handlers can be broken down into six steps.

Step one: Research. You should be familiar with the handlers and subroutines currently running on your system.

Step two: Design. Map out the functionality you plan to build, and figure out how it will run with the other handlers and subroutines.

Step three: Building. Begin creating and linking steps to perform the new functionality.

Step four: Activating. When you finish building the handler or subroutine, you must compile and move it to the IC server using the automated Publish process. Handlers and subroutines are published through Interaction Designer.

Step five: Testing. Test the handler on your live system or on a second system you've set up for testing purposes.

Step six: Promoting. If you are not already using the handler on your live system, activate the handler.

Step One: Research

The first step in building handlers and subroutines is research. You must be familiar with the handlers and subroutines currently being used on your CIC server. You need to know if the handler or subroutine you're planning can be integrated into a currently running handler. Also, find out if you need to build a new handler or subroutine, or modify an existing one. You cannot build a new handler or subroutine if you don't spend some time looking at the existing handlers.

There are several places you can look for information on the existing handlers. First, look at the contents of the Handler Help menu. Many of the handlers and subroutines shipped with CIC are documented under Handler Help. This documentation summarizes the functionality of the handlers and subroutines, and can help you see where you might want to make customizations.

Also, review the reference and procedural topics in this online help system. The tool documentation in the Reference section can help you learn the function of each step in a handler or subroutine. The procedural topics and various introductions can help you understand how handlers work.

Step Two: Planning

Once you are familiar with how the handlers and subroutines are working on the CIC server, begin planning how to build in the new functionality. Decide how the new functionality will be implemented. Review the following scenarios when making your decision.

When should I modify an existing handler or subroutine?

In many cases, modifying an existing handler is the best way to implement new functionality. In a simple example, you might want to add a new key-press option for callers using your IVR (Interactive Voice Response) system. In this case, adding new functionality only involves recording a new prompt, adding a condition to a Selection step, and adding a few other steps to your existing IVR handler. You don't need to write a whole new subroutine or handler to add this simple functionality. If you only need small changes or additions to the CIC functionality, consider small modifications to your handlers or subroutines.

If you decide to modify an existing handler or subroutine, we recommend that you only do so using the [customization points](#) provided. Customization points ease future upgrades. Modifications made outside these customization points may be overwritten when a hotfix or service release is installed. Modifying handlers outside the designated customization points may also cause undesired behavior in CIC.

When should I build a new handler or subroutine?

If you are planning a significant addition or change in functionality, you probably need to build a new handler or subroutines. In most cases, you should build a subroutine. A handler will start when an event occurs on the server to start that handler. Most of the

events that occur on the CIC server already have a handler associated with them. Unless you are adding third-party products that generate their own unique events, you build a subroutine.

An example of a piece of third-party equipment that might generate a unique event is some kind of machine monitor. Suppose a hospital has a computer monitoring a piece of equipment. Anytime that equipment fails, the monitoring computer creates an alarm event. This event could be passed into CIC and start a handler designed to watch for an alarm event.

Whether you decide to build a handler or a subroutine, map out the new functionality on a whiteboard. Try to think in terms of the Interaction Designer tools, and what groups of tools create the needed logic. Remember that creating handlers is programming; planning at the outset saves time when you start building the handler.

Step Three: Building

Once you've mapped out the functionality and decided to implement the functionality, you're ready to start modifying or building your handler or subroutines in Interaction Designer. Use Interaction Designer to add new steps and modify existing steps. For help on a particular step, click the help button on its properties page, or you can select a tool on the tool palette and press the F1 key. The help for each tool consists of a general description of the step, its parameters, and its exits. You can also find help on working with steps from the contents page of the online help. Finally, look at how steps are used in other handlers and subroutines.

You do not need to deactivate a handler before you begin editing because any changes you make to a handler or subroutine do not take effect until you publish the handler or subroutine and activate that handler or subroutine using the Manage Handlers notebook.

Step Four: Activating

Once you have finished building or modifying a handler or subroutine, you need to publish it so that your changes take effect. Publishing a handler is the automated process of converting the handler to Java code, compiling that code with a Java compiler, and placing the compiled code on the CIC server. During the publishing process, you may be notified of errors in the handler or subroutine. These errors might result from unfilled parameters, invalid variable names, or other problems. Refer to the publishing online help for advice on successful publishing. You may also need to consult the documentation that accompanied the Java compiler.

Subroutines must be published before they can be called from other handlers and subroutines. Once you have published a new subroutine, a new subroutine tool appears on your subroutine palette. Use this tool to create a new step in the handler or subroutine that calls the new subroutine.

There are certain situations where you may not want to publish your handlers or subroutines to the same server where you created them. To accomplish this, an [intermediate publish](#) file is generated and saved to a specified location, where it can then be moved to the desired server and the publishing process completed.

When a handler or subroutine is first published, you have the option to have it made active immediately. If you choose not to do this, or if the handler has been imported from another server or previously made inactive, you will need to activate it in the Manage Handlers notebook before it can be used in CIC.

Step Five: Testing

After you've published your handler or subroutine, either to a test server or the production server, test the functionality you've built.

Step Six: Promoting

If you've not already set a handler to active using the [Manage Handler notebook](#), you should do so. This activates the new functionality.

Related Topics

[Handler Best Practices](#)

[Publish a handler or subroutine](#)

Tools

Each tool has the ability to perform a specific action in a handler. For example, the Assignment tool creates a new variable and assigns a value to that variable. When a tool is dragged from the [Tools page of the Design palette](#) into a handler or subroutine, it becomes a step in that handler. As you [link](#) these steps, you create and order the series of actions that become the functionality of the handler or subroutine.

You can view the help for a specific tool by selecting that tool on the Tools page of the Design palette and pressing the F1 key, or by selecting a step in a handler or subroutine, and pressing the F1 key.

Each tool has properties that you can configure. Double-clicking a tool in the layout opens the property sheet for that tool. Each property sheet has a General page where you can give the tool a new name and write a description of that tool's purpose in the handler. Each tool might also have an Inputs page and/or an Outputs page. You can configure the way a tool behaves by changing the parameters on the Inputs and Outputs pages.

Note: A few of the tools have notebook pages with labels other than Inputs or Outputs. If you encounter one of these tools, see the documentation for that tool for more information on how to configure that tool.

The Tools page of the Design palette organizes related tools into categories. For example, all of the tools related to opening, closing, and retrieving data from a database are grouped together under the Database category. The Internet category contains tools for setting up chat sessions and sending HTML documents. Each group of tools is described in that tool category's introduction. You can access these introductions from the Contents page of this online help system.

See Also

[Power Tools](#)

[Initiators](#)

ACD

Introduction to ACD Tools

This topic describes the ACD (Automatic Communication Distribution) features of CIC. Specifically, it defines ACD in CIC, describes the parameters CIC monitors to make ACD work, and the ACD processing flow. Before you begin configuring CIC's ACD, be sure to read the *CIC ACD Processing Technical Reference* located in the PureConnect Documentation Library. This technical reference also contains information about how to configure email for a custom workgroup.

ACD in CIC

In CIC, ACD is the intelligent routing of interactions (like calls and chat sessions) to available agents in Workgroup queues. CIC uses assigned skill requirements to intelligently route incoming interactions to a qualified, available agent. Agents are assigned skill levels and other attributes in Interaction Administrator.

CIC can route calls based on the following:

- Agent skills
- Agent cost
- The amount of time an agent has been available
- Custom agent attributes that you create in Interaction Administrator
- Call priority
- The amount of time a call has been holding in a queue
- The amount of time a call has been connected in CIC

Click on one of the tools below for more information about that tool.

[ACD Agent Category](#)

[ACD Initiate Bullseye Processing](#)

[ACD Initiate Processing](#)

[ACD Interaction Details](#)

[ACD Reassign Interaction](#)

[ACD Remove Interaction Skill](#)

[ACD Select Agent](#)

[ACD Select Interaction](#)

[ACD Specify Interaction Category](#)

[ACD Specify Interaction Skill](#)

[ACD Statistics \(Interaction\)](#)

[ACD Statistics \(Queue\)](#)

[Agent Log On](#)

[Agent Log On Remote](#)

[Agent Log Off](#)

[Release Agent Lock](#)

[Release Interaction Lock](#)

[Replace Interaction](#)

[Reset In Queue Timeout](#)

[Stop ACD Processing](#)

[Workgroup Agent Activate](#)

[Workgroup Agent Deactivate](#)

ACD Agent Category

This ACD tool assigns an agent to a category. Categories allow you to create subsets within workgroups. If an agent belongs to a category and a call or chat session is assigned to that category, then only agents who are members of that category will receive that call. For example, if a call is assigned to category A (with the [ACD Specify Interaction Category](#) tool), then that call (either inbound or outbound) is assigned to an agent in category A.

Categories are less persistent than ACD skills because they are held in memory on the ACD Server. When you restart CIC, all category information is removed. You can remove an agent from a category by executing this tool with the **Add Agent to Category** option cleared. Interaction Dialer assigns agents to categories without using this tool.

Inputs

Agent Name

The name of the agent to assign to the category.

Category Name

The name of the category. If the category does not exist, this tool creates it.

Add Agent to Category

Select this option to add the agent to the specified category. Clear this option to remove the agent from the specified category.

Exit Paths

Next

This tool always takes the Next exit path.

ACD Initiate Bullseye Processing

This ACD tool starts the ACD bullseye processing for an interaction. Call this tool when an interaction is on a workgroup queue. When this tool runs, the AcServer initially considers assigning the interaction to agents in the first ring. If an agent in the first ring does not become available within the timeout period, the AcServer also considers agents in the second ring. This process continues through the five rings. If a ring does not add any additional logged on agents, the AcServer skips the ring without waiting for the timeout period to end. After consideration of all five rings, the AcServer returns to normal processing and considers any available agent in the ACD workgroup. For more information, see the *CIC ACD Processing Technical Reference* located in the PureConnect Documentation Library.

Inputs

Call Identifier

The identifier of the interaction on which ACD processing is performed.

Priority Level

The priority level assigned to the interaction. The default value is 50.

Ring 1 Agents

A list of strings specifying the names of the agents to consider for assignment in Ring 1. An agent must be logged on to be considered for assignment.

Ring 2 Agents

A list of strings specifying the names of the agents to consider for assignment in Ring 2. An agent must be logged on to be considered for assignment.

Ring 3 Agents

A list of strings specifying the names of the agents to consider for assignment in Ring 3. An agent must be logged on to be considered for assignment.

Ring 4 Agents

A list of strings specifying the names of the agents to consider for assignment in Ring 4. An agent must be logged on to be considered for assignment.

Ring 5 Agents

A list of strings specifying the names of the agents to consider for assignment in Ring 5. An agent must be logged on to be considered for assignment.

Ring 1 Timeout (sec)

The number of seconds to wait for an agent to become available in ring 1 before considering agents in ring 2. Set this value to 0 to consider agents in ring 2 immediately if no agents are available in ring 1. The default is 15.

Ring 2 Timeout (sec)

The number of seconds to wait for an agent to become available in ring 2 before considering agents in ring 3. Set this value to 0 to consider agents in ring 3 immediately if no agents are available in ring 2. The default is 15.

Ring 3 Timeout (sec)

The number of seconds to wait for an agent to become available in ring 2 before considering agents in ring 3. Set this value to 0 to consider agents in ring 3 immediately if no agents are available in ring 2. The default is 15.

Ring 4 Timeout (sec)

The number of seconds to wait for an agent to become available in ring 4 before considering agents in ring 5. Set this value to 0 to consider agents in ring 5 immediately if no agents are available in ring 4. The default is 15.

Ring 5 Timeout (sec)

The number of seconds to wait for an agent to become available in ring 5. Set this value to 0 to consider available agent in the ACD workgroup immediately if no agents are available in ring 5. The default is 15.

Ring Overflow Timeout (sec)

The number of seconds to wait after processing all rings before sending a timeout notification. The default of 0 does not raise the overflow event.

Exit Paths

Next

This step always takes the Next exit path.

ACD Initiate Processing

This ACD tool starts the ACD processing for a telephone or chat interaction. This tool should be called when an interaction is on a workgroup queue. When this step executes, the CIC server selects an appropriate agent from the workgroup queue on which the interaction currently resides. For example, if the interaction is on the Technical Support queue, then the members of the Technical Support workgroup are considered to be agents of that workgroup. All available members of that workgroup are considered agents who might receive the interaction.

In this tool, you'll also specify the weights used to calculate [Agent score](#), [interaction score](#), and other values used in ACD processing. See *Configuring ACD Processing in the CIC ACD Processing Technical Reference* located in the PureConnect Documentation Library for more information on how to configure the parameters in this tool.

Note: This tool generates a `ACDProcessQueueItem` event that starts the `ACDAvailableInteraction` handler.

Inputs

Call Identifier

The identifier of the telephone or chat interaction on which ACD processing is performed.

Weight for Agent Skills

This input is used when multiple agents are vying for one call and indicates the level of importance for Agent Skill in the Agent Score formula. Agent skill levels are assigned in Interaction Administrator on a User or Workgroup level. Specify the skill requirements for a call using one or more `ACDSpecifySkill` steps before this `ACDProcessCall` step.

If Agent Skill is more important for you than Agent Cost, Agent Available Time, or some other custom attribute, weight this parameter more than the other weight parameters. The default value is 1.0.

Weight for Agent Cost

The level of importance for Agent Cost in the Agent Score formula. Agent Cost is an attribute assigned to an agent in Interaction Administrator. Use a positive value in this parameter to assign the call to the agent with the highest cost. Use a negative value in this parameter to assign the call to the agent with the lowest cost. The default value is 1.0.

Weight for Agent Available Time

The level of importance for Agent Available Time in the Agent Score formula. If you want to assign this call to the agent who has been off the phone the longest, weight this parameter more heavily than Weight for Agent Skill and Weight for Agent Cost. The default value is 1.0.

Weight for Agent Attribute 1

Weight to be used for Agent Attribute 1. This attribute can be assigned in Interaction Administrator. The default value is 0.0.

Weight for Agent Attribute 2

Weight to be used for Agent Attribute 2. This attribute can be assigned in Interaction Administrator. The default value is 0.0.

Weight for Agent Attribute 3

Weight to be used for Agent Attribute 3. This attribute can be assigned in Interaction Administrator. The default value is 0.0.

Agent Available Time Interval

The amount of time (in seconds) before increasing the Agent Available value by one. The longer an agent is available, the higher his or her Agent score will be. The default value is 30.

Weight for Interaction Skills

If you are concerned with matching calls to agents with the best skills for the calls, then increase the Weight for Interaction Skills

more than the weights for Priority or Time in Queue or System. This input is used when an agent becomes available and there are multiple calls from which to choose. The default value is 1.0.

Weight for Priority

If you are concerned with matching calls with the highest priority to agents, then increase the Weight for Priority more than the weights for Skills or Time in Queue or System. The default value is 1.0.

Weight for Time in Queue

If you are concerned with matching calls that have been holding in this queue the longest to agents, then increase the Weight for Time in Queue more than the weights for Skills, Priority, or Time in System. The default value is 1.0.

Weight for Time in System

If you are concerned with matching calls that have been connected to CIC the longest to agents, then increase the Weight for Time in System more than the weights for Skills, Priority, or Time in Queue. The default value is 0.0.

Priority Level

The priority level assigned to the call. If you want priority taken into account when calculating interaction scores, you should also increase the value in the Weight for Priority parameter. The default value is 50.

In-Queue Time Interval

The number of seconds before the Time in Queue value is increased by one. This increases a call's score and the likelihood that it will be assigned to an agent. The default value is 30.

In-Queue Time Limit

The number of seconds before the ACD Queue Item Timeout event is generated. The [ACD Call Timeout initiator](#) is configured to start a handler when a call reaches its In-Queue time limit. Think of this parameter as the amount of time to wait before special processing is performed on the call by the ACDQueueTimeout handler. The default value is 0, which indicates no time limit.

In-System Time Interval

The amount of time (in seconds) before increasing the Time in System by value by one. This increases a call's score and the likelihood that it will be assigned to an agent. The default value is 30.

Weight for Utilization

Increase Weight for Utilization if you want agents whose utilization is consumed by ACD interactions already assigned to them to be less likely to get a new ACD interaction assigned in a scenario with multiple agents available in the queue. The default value is 0.0.

Outputs

Call Identifier

The identifier for the call flagged for ACD processing.

Exit Paths

Next

This step always takes the Next exit path.

ACD Interaction Details

This ACD tool retrieves the ACD properties assigned to a specified interaction.

Inputs

Call Identifier

The unique identifier for an interaction.

Outputs

Interaction is Acd

Whether or not the specified interaction is ACD.

Acd Interaction Priority

The calculated score for the interaction as determined by Interaction Administrator.

Interaction Time in Acd Queue

The amount of time in seconds that the interaction has been in the ACD queue.

Interaction Time Until Acd Timeout

The number of seconds remaining before the interaction times out.

Acd Item is in Queue Timeout

Boolean indicating whether or not the interaction has timed out.

Acd Interaction Is Currently On Hold

Boolean indicating whether or not the interaction is on hold.

Acd Interaction Time In System

The total number of seconds that the interaction has been in the system.

Acd Interaction Answered Timestamp

The timestamp for when the interaction was first connected.

Acd Skills required for Interaction

List of skills required for this interaction as specified in Interaction Administrator.

Acd Skill Weights for Interaction

Weights for skills pertaining to this interaction as specified in Interaction Administrator.

Acd Interaction Skill Proficiencies

The minimum skill proficiencies assigned to this interaction.

Acid Categories for Interaction

The ACD category assigned to this interaction (i.e., ACD, Custom, Group Ring, or Sequential) as specified in Interaction Administrator.

Acid Queue That Interaction is Assigned to

The ID of the queue that the interaction is currently assigned to.

Acid Interaction Assigned Agent

The User ID of the Agent assigned to this interaction.

Exit Paths

Success

This path is taken if the ACD properties are successfully retrieved.

Failure

This path is taken if the operation fails.

ACD Reassign Interaction

This ACD tool reassigns an interaction to an agent when the agent to which it was originally assigned does not answer. It is only used when an agent does not pick up a call that was assigned to them.

Inputs

Call Identifier

The identifier of the telephone or chat interaction to be reassigned.

Agent Name

This parameter is not currently recognized.

Do not assign any other calls to this agent

Set this parameter to true if you do not want any interactions assigned to the agent that was not answering for thirty seconds. This gives the agent who is answering time to finish what he or she is doing and to set his or her status to "Available."

Exit Paths

Next

This tool always takes the Next exit path.

ACD Remove Interaction Skill

Use this ACD tool when an interaction no longer requires agents with the specified skill. The specified skill will not be factored into the final interaction/agent score. If the interaction has not been assigned, it will be reevaluated against all available agents using the new score.

Inputs

Call Identifier

The unique identifier for this interaction.

Skill Name

The skill being removed from consideration.

Exit Paths

Success

This path is taken if the skill has been successfully removed.

Failure

This path is taken if the operation fails.

ACD Select Agent

This ACD tool assigns a telephone or chat interaction to an available agent in the workgroup containing the queue item. The agent selected is the agent with the highest [agent score](#).

Before you begin configuring CIC's ACD, be sure to read the *CIC ACD Processing Technical Reference* located in the PureConnect Documentation Library.

Inputs

Call Identifier

The identifier of telephone or chat interaction to assign to an agent.

Outputs

Agent for Assignment

An agent who can accept the queue item. This is the most appropriate agent as determined by the [agent score](#) or [interaction score](#).

Exit Paths

Success

The Success exit path is taken if the queue item (call or chat session) could be assigned to an agent.

No Available Agents

This path is taken if there are no agents available to take the interaction.

Already Assigned

This path is taken if the queue item is already assigned to an agent.

Failure

The Failure exit path is taken if the queue identifier is not valid.

ACD Select Interaction

This ACD tool assigns a telephone or chat interaction to an available agent. The interaction selected is the interaction with the highest [interaction score](#).

Inputs

Agent Name

The name of the agent to which the call is assigned. If the agent name is not a scoped queue name (such as "User:*Name*"), it is assumed to correspond to a user queue.

Outputs

Call Identifier

The identifier for the telephone or chat interaction being assigned.

Workgroup Queue Name

The identifier for the workgroup queue on which this call is being processed.

Exit Paths

Success

The Success exit path is taken if the queue item (call or chat session) could be assigned to an agent.

Failure

The Failure exit path is taken if the queue identifier is not valid or if the call could not be assigned.

ACD Specify Interaction Category

This ACD tool assigns a call to a category. Categories allow you to create subsets within workgroups. If an agent belongs to a category and call or chat session is assigned to that category, then only agents who are members of that category will receive that call. For example, if a call is assigned to category A, then that call (either inbound or outbound) is assigned to an agent in category A. Agents are assigned to categories with the [ACD Agent Category](#) tool.

Categories are less persistent than ACD skills because they are held in memory on the ACD Server. When you restart CIC, all category information is removed. You can remove an agent from a category by executing this tool with the ACD Agent to Category option clear. The [ACD Agent Logout](#) tool removes an agent from all categories. The Interaction Dialer application assigns agents to categories without using this tool.

Inputs

Call Identifier

The call to be assigned to a category.

Category Name

The name of the category. If the category does not exist, it is created.

Agents must be in this category

If this option is selected, one or more agents must be in the category before the call can be assigned to that category. If this option is not selected, it removes the call from the specified category.

Exit Paths

Next

This tool will always take the Next exit path.

ACD Specify Interaction Skill

This ACD tool defines a skill needed for a call. ACD Specify Interaction Skill steps precede [ACD Initiate Processing](#) steps. If you are specifying skills, you probably also want to configure the ACD Initiate Processing tool to assign the call to an agent based on skill. If this is the case, then increase the values for Weight for Agent Skill and Weight for Skill in the ACD Initiate Processing tool accordingly.

See the *CIC ACD Processing Technical Reference* located in the PureConnect Documentation Library for more information on configuring ACD skills.

Note: When ACD Server evaluates a call's skills, it first looks for skills that were explicitly assigned to that agent from Interaction Administrator. If the required skills were not explicitly assigned, ACD Server looks at the skills that the agent inherited from any workgroups to which he or she belongs. The agent's entire set of skills is used to evaluate an interaction on a workgroup queue.

Reassigning Skills

In some cases you may want to reassign the skill levels required for a call. For example, if a call has timed out because no agents with the necessary skills were available, then you may want to lower the minimum proficiency levels. All you have to do is run the call through another ACD Specify Interaction Skill step with the new minimum proficiency levels. It is necessary to call [ACD Initiate Processing](#) when the skills have changed. Also, after changing skill levels, a [Set Call State](#) tool must be set with the "cancel pending operations" option selected in order to cancel all prompts that were playing at the time the skills were changed.

Inputs

Call Identifier

The identifier for the interaction for which you want to specify skills.

Skill Name

The skill name as specified in Interaction Administrator.

Minimum Proficiency Level

The minimum proficiency level required for this skill. Agents who do not possess this minimum skill proficiency will not be eligible to receive this interaction.

Maximum Proficiency Level

The maximum proficiency level required for this skill. Agents who exceed this maximum skill proficiency will not be eligible to receive this interaction.

Weight for Proficiency Level

How important Proficiency Level is for this skill, as opposed to other skills you specify in other ACDSpecifySkill steps. If this is the only skill you are specifying, then leave the default value in this parameter. If you are more concerned with this skill, then weight this parameter more heavily than Weight for Proficiency Level in other ACDSpecifySkill steps in this handler.

Minimum Desire to Use Level

The value for Desire to Use a particular skill is assigned on an agent by agent bases in Interaction Administrator. Agents who do not meet the minimum Desire to Use level you specify in this step are not eligible to receive this interaction.

Maximum Desire to Use Level

The value for Desire to Use a particular skill is assigned on an agent by agent bases in Interaction Administrator. Agents who exceed the maximum Desire to Use level you specify in this step are not eligible to receive this interaction.

Weight for Desire to Use

How important Desire to Use this skill is, as opposed to other skills you specify in other ACDSpecifySkill steps. If this is the only skill you are specifying, then leave the default value in this parameter. If you are more concerned with this skill, then weight this parameter more heavily than Weight for Desire to Use in other ACDSpecifySkill steps in this handler.

Exit Paths

Next

This step always takes the Next exit path.

ACD Statistics (Interaction)

This ACD tool returns statistics for reporting on ACD interactions.

Inputs

Call Identifier

The identifier for the call on which to gather statistics.

Outputs

Acid Queue containing this call

The scoped workgroup queue name for the workgroup containing the call.

Estimated wait time

An estimate of the amount of time (in seconds) that a call will wait before being connected to an agent. This wait time is calculated using the average wait time and subtracting the amount of time that the call has been on the queue. Average wait time is calculated by adding the total wait time for all calls answered in a given period (the previous 30 minutes) and dividing that total by the number of calls answered in that period. A negative value is returned if the average wait time statistics are not available (due to insufficient observations in the sampling period) or if the call has already waited longer than the estimated wait time.

Estimated position in queue

The number of calls ahead of this call. -1 means this call is currently assigned, 1 means this call will be answered next, 2 means this is the second call to be answered, and so on.

Number of Agents Logged In

The number of agents (whose skills match the call's requirements) currently logged into this queue.

Number of Agents Available

The number of agents whose skills match the call's requirements and who are 100% available to take calls.

Note: If an agent is on any interaction (regardless of utilization), he or she will show as unavailable.

Number of Agents on Calls

The number of agents (whose skills match the call's requirements) currently connected to a call.

Number of Agents in Follow Up

The number of agents (whose skills match the call's requirements) whose status is set to a Follup Up status.

Longest Time an Agent has been Available

The longest time (in seconds) that an agent has been available. This is calculated for all logged-in agents, regardless of skills.

Number of calls waiting assignment

The number of calls in this queue waiting to be assigned to an available agent.

Number of Connected Calls

The number of calls currently connected to an agent.

Longest Wait Time for a Call

The amount time (in seconds) that the longest waiting call has been waiting.

Average Wait Time for Connected Calls

The average wait time (in seconds) for connected calls.

Average Wait Time for Abandoned Calls

The average wait time (in seconds) for calls that disconnected before connecting.

Average Duration of Connected Calls

The average time (in seconds) that all connected calls have been connected.

Average Follow Up Time for Connected Calls

The average follow up time (in seconds) for calls that were connected to an agent.

Denominator for Average Wait Time

The number of calls used to calculate the average wait time.

Denominator for Average Abandon Time

The number of calls used to calculate the average abandon time.

Denominator for Average Duration

The number of calls used to calculate the average duration.

Denominator for Average Follow Up Time

The number of calls used to calculate the average follow up time.

Exit Paths

Success

This tool takes the Success exit path if the call ID is valid and the tool encountered no errors.

Failure

This step always takes the Failure exit path if the call ID is no longer valid, if there is a catastrophic problem on the CIC Server,

ACD Statistics (Queue)

This ACD tool returns statistics used for reporting on workgroup queues containing ACD calls.

Inputs

Workgroup Queue Name

The name of the queue on which you want to collect statistics.

Outputs

Estimated wait time

An estimate of the amount of time (in seconds) that a call will wait before being connected to an agent. This wait time is calculated using the average wait time for the queue and subtracting the amount of time that the call has been on the queue. Average wait

time is calculated by adding the total wait time for all calls answered in a given period (the previous 30 minutes) and dividing that total by the number of calls answered in that period. A negative value is returned if the average wait time statistics are not available (due to insufficient observations in the sampling period) or if the call has already waited longer than the estimated wait time.

Number of Agents Logged In

The number of agents logged into this queue.

Number of Agents Available

The number of agents who are 100% available to take ACD calls on this queue.

Note: If an agent is on any interaction (regardless of utilization), he or she will show as unavailable.

Number of Agents on Calls

The number of agents currently connected to calls.

Number of Agents in Follow Up

The number of agent's whose status is set to a Follow Up status.

Longest Time an Agent has been Available

The longest time (in seconds) an agent has been available. This is calculated for all logged-in agents.

Number of calls waiting assignment

The number of ACD calls in this queue that have not been assigned to agents.

Number of Connected Calls

The number of call currently connected to an agent.

Longest Wait Time for a Call

The longest amount of time (in seconds) a call waited before being connected.

Average Wait Time for Connected Calls

The average amount of time (in seconds) that calls waited before being connected.

Average Wait Time for Abandoned Calls

The average amount of time (in seconds) that calls waited before abandoning.

Average Duration of Connected Calls

The average amount of time (in seconds) that connected calls remained connected.

Average Follow Up Time for Connected Calls

The average amount of time (in seconds) that agents were in a follow up status after completing a call.

Denominator for Average Wait Time

The number of calls used to calculate average wait time.

Denominator for Average Abandon Time

The number of calls used to calculate average abandon time.

Denominator for Average Duration

The number of calls used to calculate average duration.

Denominator for Average Follow Up Time

The number of calls used to calculate average follow up time.

Exit Paths

Success

This tool takes the Success exit path if the queue name is valid and the tool encountered no errors.

Failure

This step always takes the Failure exit path if the call ID is no longer valid, if there is a catastrophic problem on the CIC Server.

Agent Log On

This ACD tool logs an agent in so that he or she may receive calls. It is useful for agents who have stations but are not running a CIC client.

Inputs

Agent Name

The name of the agent to be logged in.

Workstation ID

The workstation where calls for the specified agent will be sent.

Exit Paths

Success

This tool takes the Success exit path if the Agent Name and the Workstation ID are valid.

Failure

This tool takes the Failure exit path if the Agent Name and the Workstation ID are invalid.

Conditional Success - No ACD

This tool takes the Conditional Success - No ACD exit path even if ACD agents and queues are configured properly but there was a license failure. A license failure could mean there were not enough ACD, ACD1, or ACD2 types of licenses available for the agent to log in. Check the license availability in Interaction Administrator.

Agent Log On Remote

This ACD tool logs an agent in from a remote location so that he or she may receive calls.

Note: This tool acts like a client remote login. By default, if an agent is not configured for auto-answer ACD and non-ACD calls, their calls will not alert their phone. Only auto-answer calls force a connect call to go out to remote login agents.

Inputs

Agent Name

The CIC user name of the agent to be logged in.

Remote Number for Agent

The number from where the agent is calling.

Persistent Connection Flag

Set this Boolean to True to maintain a persistent voice connecton to the CIC Server. The audio path will not disconnect until a disconnect is initiated at that station.

If left False, CIC will determine when the audio path to the station is no longer needed, and initiate the disconnect automatically.

Exit Paths

Success

This tool takes the Success exit path if the Agent Name and the Remote Number are valid.

Failure

This tool takes the Failure exit path if the Agent Name and/or the Remote Number are invalid.

Conditional Success - No ACD

This tool takes the Conditional Success - No ACD exit path even if ACD agents and queues are configured properly but there was a license failure. A license failure could mean there were not enough ACD, ACD1, or ACD2 types of licenses available for the agent to log in. Check the license availability in Interaction Administrator.

Agent Log Off

This ACD tool logs an agent out so that he or she will receive no further calls. This tool is useful for agents who have stations but are not running a CIC client.

Inputs

Agent Name

The name of the agent to be logged out.

Workstation ID

The workstation where calls for the specified agent were sent.

Exit Paths

Success

This tool takes the Next exit path, unless the workstation ID is invalid.

Failure

This tool takes the Failure exit path if the workstation ID is invalid.

Release Agent Lock

This ACD tool, along with its counterpart [Release Interaction Lock](#), is seldom needed unless you customize the default ACD handlers or create new handlers for special ACD processing. This is because the default handlers appropriately release locks on agents. They are normally locked between the time an ACD agent becomes available until after the ACD Select Agent tool step completes. Custom handlers that do not use the ACD Select Agent tool step will not release the lock, which prevents the agent from being assigned an interaction. In this case, use the Release Agent Lock tool to release a locked agent in a timely manner. If you do not use this tool, the agent lock will eventually time out (after 10 seconds by default), but it may cause problems when assigning ACD interactions to agents.

You can specify a different default timeout period for CIC to release agent locks. To do this, create a server parameter named `ACDAgentLockTimeout` and assign it an integer that specifies the number of seconds the system will wait before automatically releasing an agent lock. To have more precise control over this process, use the Release Agent Lock tool instead of relying on this server parameter.

Inputs

Agent Name

The CIC name of the agent currently locked.

Exit Paths

Next

This tool always takes the Next exit path.

Release Interaction Lock

This ACD tool, along with its counterpart [Release Agent Lock](#), is seldom needed unless you customize the default ACD handler or create new handlers for special ACD processing. This is because the default handlers appropriately release locks on interactions. They are normally locked between the ACD Process Item tool step until after the ACD Select Item tool step completes. Custom handlers that do not use the ACD Select Item tool step will not release the lock, which prevents the interaction from being assigned. In this case, use the Release Interaction Lock tool to release a locked interaction in a timely manner. If you do not use this tool, the interaction lock will eventually time out (after 10 seconds by default).

You can specify a different default timeout period for CIC to release interaction locks. To do this, create a server parameter named **ACDInteractionLockTimeout** and assign it an integer that specifies the number of seconds the system will wait before automatically releasing an interaction lock. To have more precise control over this process, use the Release Interaction Lock tool instead of relying on this server parameter.

Inputs

Call Identifier

The identifier ("CallID") for the interaction to be unlocked. This is not limited to ACD calls but can be any type of ACD interaction.

Exit Paths

Next

This tool always takes the Next exit path.

Replace Interaction

This ACD tool takes a telephone call that is undergoing ACD processing, and replaces it with a specified object that is on the system queue. This tool is intended to replace a telephone call with a callback object. Only telephone calls may be replaced with this tool, though any object type may be used for the replacement object.

After insuring that the original call is in ACD processing and that it is not disconnected, the tool removes the original interaction from all the queues and inserts the specified (callback) interaction in all the queues the original interaction was in. After being replaced, the original call is removed from the workgroup queue and disconnected if the disconnect flag is set to true.

Note: Base attributes, such as time in queue, skills, priority, etc., will need to be carried by the handler that is responsible for creating the callback.

You may lose access to attributes of the original call interaction, so it's recommended that you save a copy of necessary attribute values within the handler prior to using this toolstep.

Inputs

Call ID of Original Interaction

Unique identifier of the interaction to be replaced.

Call ID of Replacement Interaction

Unique identifier of the replacement interaction.

Disconnect Original Interaction

If true, disconnects the replaced interaction.

Exit Paths

Success

This path is taken if the call is successfully replaced.

Failure

This path is taken if the operation fails. Failure can occur for the following reasons.

1. The original interaction is not undergoing ACD processing.
2. The original interaction has already been assigned.
3. The original interaction has disconnected.
4. The original interaction is not a telephone call.
5. The replacement object is on a user or workgroup queue.

Reset In Queue Timeout

This ACD tool resets the timeout previously set for a call after it has begun ACD for processing. This can be used, for example, to disable the timeout for a caller who has opted to go to an IVR during ACD wait. When a call's timeout limit is reset, any time already spent in the queue is discarded.

Using this tool after an in-queue timeout will not cause the ACD server to assign the call. If you want to assign a call that has already been timed out, you must use the [ACD Initiate Processing](#) tool again.

Inputs

Call Identifier

The identifier of the telephone or chat interaction on which ACD processing is being performed.

New In-Queue Time Limit (Seconds)

This sets the new timeout period for the call. Setting the new limit to 0 (zero) will allow the call to remain in-queue indefinitely.

Exit Paths

Success

The Success exit path is taken if the queue timeout is successfully reset.

Failure

The Failure exit path is taken if the operation fails.

Stop ACD Processing

This ACD tool stops the ACD processing for a telephone or chat interaction.

Inputs

Call Identifier

The identifier of the telephone or chat interaction on which ACD processing is being performed.

Exit Paths

Success

The Success exit path is taken if the ACD Processing is successfully stopped.

Failure

The Failure exit path is taken if the operation fails.

Workgroup Agent Activate

This ACD tool activates an agent in a workgroup. The agent must first be a member of the target workgroup queue. If the agent is a member of more than one workgroup, the agent's activation status remains unchanged for all workgroups except the one specified.

Note: This tool was formerly (in IC 2.3x) named ACD Queue Login.

Inputs

Agent Name

The CIC user name of the agent to activate.

Workgroup Name

The name of the workgroup queue in which to activate the agent.

Exit Paths

Success

This path is taken if the specified agent is successfully activated in the specified queue.

Failure

This path is taken if the agent could not be activated. This happens if the agent is not a member of the specified workgroup queue.

Workgroup Agent Deactivate

This ACD tool deactivates an agent from a workgroup queue in which he or she is a member. If the agent belongs to more than one workgroup queue, his or her activation status will remain unchanged in all others except the one specified.

Note: This tool was formerly (in IC 2.3x) named ACD Queue Logout.

Inputs

Agent Name

The name of the agent to be deactivated.

Workgroup Name

The name of the workgroup queue the agent is deactivating.

Exit Paths

Success

This path is taken if the specified agent is successfully deactivated from the specified queue.

Failure

This path is taken if the agent could not be deactivated for any reason. This happens if the agent is not a member of the specified workgroup queue.

Accumulators

Accumulators

Accumulators are generic, on-the-fly, global variables. The accumulator tools on the tool palette allow the user to create and modify the behavior of accumulators and the types of information they accumulate. Accumulators provide a place to store an attribute. For example, accumulators provide a way for someone to gather (accumulate) a total number of calls. Accumulators are defined in Interaction Administrator. Interaction Designer users can decide when they want to gather the information that is stored in the accumulators.

While accumulators are defined in Interaction Administrator, the accumulator tools create instances of accumulators and modify the values of those instances.

Accumulator Tool Actions

Accumulators are not values themselves. They are definitions of types of data to be collected. An instance of an accumulator is an actual value being saved with the system. The value of the instance may be set, retrieved, incremented, or locked. These actions are described in the following list:

- **Set:** They set the value of an instance (one for each data type: Boolean, DateTime, Integer, Number, String).
- **Increment:** They increment or decrement the value of an instance (Integer or Number type only).
- **Get:** They get a value from the accumulator and bring it to the handler (one for each data type: Boolean, DateTime, Integer, Double, String).
- **Lock/Unlock:** They allow/deny access to the value of an instance. (One for each data type: Boolean, DateTime, Integer, Number, String). Locking is an advanced feature that might be needed if a new value for an accumulator must be calculated using multiple steps. There is a chance that a "Lost Update" could occur if another handler is allowed to change the accumulator value while these multiple steps are executing.

Related Topics

[Get Acc. \(Boolean\)](#)

[Get Acc. \(Datetime\)](#)

[Get Acc. \(Integer\)](#)

[Get Acc. \(Numeric\)](#)

[Get Acc. \(String\)](#)

[Inc. Acc. \(Integer\)](#)

[Inc. Acc. \(Numeric\)](#)

[Lock Accumulator](#)

[Set Acc \(Boolean\)](#)

[Set Acc \(Datetime\)](#)

[Set Acc. \(Integer\)](#)

[Set Acc. \(Numeric\)](#)

[Set Acc. \(String\)](#)

[Unlock Accumulator](#)

Get Acc. (Boolean)

This Accumulator step gets the value of an instance of an accumulator of Boolean type.

Inputs

Accumulator Name

The name of the accumulator.

Instance Name

The Instance Name is defined in Interaction Administrator and created dynamically when a handler executes a Set Acc. step.

Outputs

Boolean Accumulator Value Retrieved

The variable specified here contains the value of the accumulator instance.

Exit Paths

Success

This step takes the Success exit path if the operation is successful.

Failure

This step takes the Failure exit path if the Instance Name is invalid for the accumulator. This would never happen if the Instance type was Any, but it is possible for the other instance types. Failure can also occur if communication between the IP process and the Accumulator server timed out due to heavy machine load.

Get Acc. (DateTime)

This Accumulator step gets the value of an instance of an accumulator of DateTime type.

Inputs

Accumulator Name

The name of the accumulator.

Instance Name

The Instance Name is defined in Interaction Administrator and created dynamically when a handler executes a Set Acc step.

Outputs

DateTime Accumulator Value Retrieved

The variable specified here contains the value of the accumulator instance.

Exit Paths

Success

This step takes the Success exit path if the operation is successful.

Failure

This step takes the Failure exit path if the Instance Name is invalid for the accumulator. This would never happen if the Instance type was Any, but it is possible for the other instance types. Failure can also occur if communication between the IP process and the Accumulator server timed out due to heavy machine load.

Get Acc. (Integer)

This Accumulator step gets the value of an instance of an accumulator of Integer type.

Inputs

Accumulator Name

The name of the accumulator as defined in Interaction Administrator.

Instance Name

The Instance Name is defined in Interaction Administrator and created dynamically when a handler executes a Set Acc step.

Outputs

Integer Accumulator Value Retrieved

The variable specified here contains the value of the accumulator instance.

Exit Paths

Success

This step takes the Success exit path if the operation is successful.

Failure

This step takes the Failure exit path if the Instance Name is invalid for the accumulator. This would never happen if the Instance type was Any, but it is possible for the other instance types. Failure can also occur if communication between the IP process and the Accumulator server timed out due to heavy machine load.

Get Acc. (Numeric)

This Accumulator step gets the value of an instance of an accumulator of Numeric type.

Inputs

Accumulator Name

The name of the accumulator as defined in Interaction Administrator.

Instance Name

The Instance Name is defined in Interaction Administrator and created dynamically when a handler executes a Set Acc step.

Outputs

Double Accumulator Value Retrieved

The variable specified here contains the value of the accumulator instance.

Exit Paths

Success

This step takes the Success exit path if the operation is successful.

Failure

This step takes the Failure exit path if the Instance Name is invalid for the accumulator. This would never happen if the Instance type was Any, but it is possible for the other instance types. Failure can also occur if communication between the IP process and the Accumulator server timed out due to heavy machine load.

Get Acc. (String)

This Accumulator step gets the value of an instance of an accumulator of String type.

Note: String accumulator values are limited to 254 characters. Longer strings will be truncated.

Inputs

Accumulator Name

The name of the accumulator as defined in Interaction Administrator.

Instance Name

The Instance Name is defined in Interaction Administrator and created dynamically when a handler executes a Set Acc step.

Outputs

String Accumulator Value Retrieved

The variable specified here contains the value of the accumulator instance.

Exit Paths

Success

This step takes the Success exit path if the operation is successful.

Failure

This step takes the Failure exit path if the Instance Name is invalid for the accumulator. This would never happen if the Instance type was Any, but it is possible for the other instance types. Failure can also occur if communication between the IP process and the Accumulator server timed out due to heavy machine load.

Inc. Acc. (Integer)

This Accumulator step increments the value of an instance of an accumulator of type Integer.

Caution: Integer Accumulator values must fall between -2,147,483,648 and 2,147,483,647.

Inputs

Accumulator Name

The name of the accumulator to get the value from. This name is defined in Interaction Administrator.

Instance Name

The specific instance of the accumulator you want to increment. The Instance Name is defined in Interaction Administrator and created dynamically when a handler executes a [Set Acc. \(Integer\)](#) step.

Value

The value by which to increment the instance of the accumulator. Entering a negative number decrements the value of the instance of the accumulator.

Accumulator Key

This is the variable that holds the value of the key returned by a previous Lock Accumulator step. The Accumulator Key is not a required input field, but should be supplied if the accumulator instance has been previously locked by the handler. Failure to use a key when the accumulator instance is locked can result in a tool step failure, and incorrect values being calculated in the accumulator.

Exit Paths

Success

This step takes the Success exit path if the operation is successful.

Failure

This step takes the Failure exit path if the Instance Name is invalid for the accumulator. This would never happen if the Instance type was Any, but it is possible for the other instance types. Failure can also occur if communication between the Interaction Processor process and the Accumulator server timed out due to heavy machine load. Failure to use a key when the accumulator instance is locked can result in a tool step failure and incorrect values being calculated in the accumulator.

Inc. Acc. (Numeric)

This Accumulator step increments the value of an instance of an accumulator of type Numeric.

Caution: Numeric Accumulator values are limited to 1.7E +/- 308 (15 digits precision).

Inputs

Accumulator Name

The name of the accumulator to get the value from. This name is defined in Interaction Administrator.

Instance Name

The specific instance of the accumulator you want to increment. The Instance Name is defined in Interaction Administrator and created dynamically when a handler executes a Set Acc. step.

Value

The value by which to increment the instance of the accumulator. Entering a negative number decrements the value of the instance of the accumulator.

Accumulator Key

This is the variable that holds the value of the key returned by a previous Lock Accumulator step. The Accumulator Key is not a required input field, but should be supplied if the accumulator instance has been previously locked by the handler. Failure to use a key when the accumulator instance is locked can result in a tool step failure, and incorrect values being calculated in the accumulator.

Exit Paths

Success

This step takes the Success exit path if the operation is successful.

Failure

This step takes the Failure exit path if the Instance Name is invalid for the accumulator. This would never happen if the Instance type was Any, but it is possible for the other instance types. Failure can also occur if communication between the IP process and the Accumulator server timed out due to heavy machine load. Failure to use a Key when the accumulator instance is locked can result in a tool step failure, and incorrect values being calculated in the accumulator.

Lock Accumulator

This Accumulator step locks an instance of an accumulator so that the value does not change. This step prevents another accumulator tool from changing a value before that value can be retrieved. A lock will eventually expire so that other tools can get or change the value of an accumulator instance.

Inputs

Accumulator Name

The name of the accumulator to lock. This name is defined in Interaction Administrator.

Instance Name

The specific instance of the accumulator you want to lock. The rules for this instance name are defined in Interaction Administrator.

Outputs

Accumulator Key

Contains the value of the key. This key must be used by tools that access this accumulator instance. This key must also be provided to an Unlock Accumulator step.

Exit Paths

Success

This step takes the Success exit path if the operation is successful.

Failure

This step takes the Failure exit path if the Instance Name is invalid for the accumulator not in the list of valid instances. This would never happen if the Instance type was Any, but it is possible for the other instance types. Failure can also occur when communication between the IP process and the Accumulator server timed out due to heavy machine load.

Set Acc. (Boolean)

This Accumulator step assigns a value to an instance of an Accumulator of type Boolean. This step can also create a new instance of an accumulator.

Inputs

Accumulator Name

The name of the accumulator to get the value from. This name is defined in Interaction Administrator.

Instance Name

The specific instance of the accumulator you want to assign a value to. The rules for this instance name are defined in Interaction Administrator, and if the value of this parameter is unique, a new instance is created when this step executes.

Value

The value to assign to the instance of the accumulator.

Accumulator Key

This is the variable that holds the value of the key returned by a previous Lock Accumulator step. The Accumulator Key is not a required input field, but should be supplied if the accumulator instance has been previously locked by the handler. Failure to use a Key when the accumulator instance is locked can result in a tool step failure, and incorrect values being calculated in the accumulator.

Exit Paths

Success

This step takes the Success exit path if the operation is successful.

Failure

This step takes the Failure exit path if the Instance Name is invalid for the accumulator. This would never happen if the Instance type was Any, but it is possible for the other instance types. Failure can also occur if communication between the IP process and the Accumulator server timed out due to heavy machine load. Failure to use a Key when the accumulator instance is locked can result in a tool step failure, and incorrect values being calculated in the accumulator.

Set Acc. (Datetime)

This Accumulator step assigns a value to an instance of an Accumulator of type Datetime. This step can also create a new instance of an accumulator.

Inputs

Accumulator Name

The name of the accumulator to get the value from. This name is defined in Interaction Administrator.

Instance Name

The specific instance of the accumulator to which you want to assign a value. The rules for this instance name are defined in Interaction Administrator, and if the value of this parameter is unique, a new instance is created when this step executes.

Value

The value to assign to the instance of the accumulator.

Accumulator Key

This is the variable that holds the value of the key returned by a previous [Lock Accumulator](#) step. The Accumulator Key is not a required input field, but should be supplied if the accumulator instance has been previously locked by the handler. Failure to use a Key when the accumulator instance is locked can result in a tool step failure, and incorrect values being calculated in the accumulator.

Exit Paths

Success

This step takes the Success exit path if the operation is successful.

Failure

This step takes the Failure exit path if the Instance Name is invalid for the accumulator. This would never happen if the Instance type was Any, but it is possible for the other instance types. Failure can also occur if communication between the IP process and the Accumulator server timed out due to heavy machine load. Failure to use a Key when the accumulator instance is locked can result in a tool step failure, and incorrect values being calculated in the accumulator.

Set Acc. (Integer)

This Accumulator step assigns a value to an instance of an accumulator of type Integer. This step can also create a new instance of an accumulator.

Caution: Integer Accumulator values must fall between -2,147,483,648 and 2,147,483,647.

Inputs

Accumulator Name

The name of the accumulator from which you want to get the value. This name is defined in Interaction Administrator.

Instance Name

The specific instance of the accumulator you want to assign a value. The rules for this instance name are defined in Interaction Administrator. If the value of this parameter is unique, a new instance is created when this step executes.

Value

The value to assign to the instance of the accumulator.

Accumulator Key

This is the variable that holds the value of the key returned by a previous Lock Accumulator step. The Accumulator Key is not a required input field, but should be supplied if the accumulator instance has been previously locked by the handler. Failure to use a Key when the accumulator instance is locked can result in a tool step failure, and incorrect values being calculated in the accumulator.

Exit Paths

Success

This step takes the Success exit path if the operation is successful.

Failure

This step takes the Failure exit path if the Instance Name is invalid for the accumulator. This would never happen if the Instance type was Any, but it is possible for the other instance types. Failure can also occur if communication between the IP process and the Accumulator server timed out due to heavy machine load. Failure to use a Key when the accumulator instance is locked can result in a tool step failure, and incorrect values being calculated in the accumulator.

Set Acc. (Numeric)

This Accumulator step assigns a value to an instance of an accumulator of type Numeric. This step can also create a new instance of an accumulator.

Caution: Numeric Accumulator values are limited to 1.7E +/- 308 (15 digits precision).

Inputs

Accumulator Name

The name of the accumulator to get the value from. This name is defined in Interaction Administrator.

Instance Name

The specific instance of the accumulator you want to assign a value to. The rules for this instance name are defined in Interaction Administrator, and if the value of this parameter is unique, a new instance is created when this step executes.

Value

The value to assign to the instance of the accumulator.

Accumulator Key

This is the variable that holds the value of the key returned by a previous Lock Accumulator step. The Accumulator Key is not a required input field, but should be supplied if the accumulator instance has been previously locked by the handler. Failure to use a Key when the accumulator instance is locked can result in a tool step failure, and incorrect values being calculated in the accumulator.

Exit Paths

Success

This step takes the Success exit path if the operation is successful.

Failure

This step takes the Failure exit path if the Instance Name is invalid for the accumulator. This would never happen if the Instance type was Any, but it is possible for the other instance types. Failure can also occur if communication between the IP process and the Accumulator server timed out due to heavy machine load. Failure to use a Key when the accumulator instance is locked can result in a tool step failure, and incorrect values being calculated in the accumulator.

Set Acc. (String)

This Accumulator step assigns a value to an instance of an Accumulator of type string. This step can also create a new instance of an accumulator.

Note: String accumulator values are limited to 254 characters. Longer strings will be truncated.

Inputs

Accumulator Name

The name of the accumulator to get the value from. This name is defined in Interaction Administrator.

Instance Name

The specific instance of the accumulator to which you want to assign a value. The rules for this instance name are defined in Interaction Administrator, and if the value of this parameter is unique, a new instance is created when this step executes.

Value

The value to assign to the instance of the accumulator.

Accumulator Key

This is the variable that holds the value of the key returned by a previous Lock Accumulator step. The Accumulator Key is not a required input field, but should be supplied if the accumulator instance has been previously locked by the handler. Failure to use a Key when the accumulator instance is locked can result in a tool step failure, and incorrect values being calculated in the accumulator.

Exit Paths

Success

This step takes the Success exit path if the operation is successful.

Failure

This step takes the Failure exit path if the Instance Name is invalid for the accumulator. This would never happen if the Instance type was Any, but it is possible for the other instance types. Failure can also occur if communication between the IP process and the Accumulator server timed out due to heavy machine load. Failure to use a Key when the accumulator instance is locked can result in a tool step failure, and incorrect values being calculated in the accumulator.

Unlock Accumulator

This Accumulator step unlocks an instance of an accumulator so that the value can be changed. This step reverses the effect of a Lock Accumulator step.

Inputs

Accumulator Name

The name of the accumulator to unlock. This name is defined in Interaction Administrator.

Instance Name

The specific instance of the accumulator you want to unlock.

Accumulator Key

Contains the value of the key. This key was created with a Lock Accumulator step.

Exit Paths

Success

This step takes the Success exit path if the operation is successful.

Failure

This step takes the Failure exit path if the Instance Name is invalid for the accumulator. This would never happen if the Instance type was Any, but it is possible for the other instance types. Failure can also occur if communication between the IP process and the Accumulator server timed out due to heavy machine load. Failure to use a Key when the accumulator instance is locked can result in a tool step failure, and incorrect values being calculated in the accumulator.

Basic

Introduction to Basic Tools

The Basic tools are versatile tools that provide common programming functionality to the handlers. They are used to create variables, assign values to variables, and evaluate the values of variables.

Click on one of the tools below for more information about that tool.

[Assignment](#)

[Condition](#)

[Notify Debugger](#)

[Selection](#)

[Table Lookup](#)

[Write Trace Message](#)

Assignment Tool

This Basic tool declares a variable and assigns a [value](#) to that new variable or to an existing variable. The assigned value can be the value of another variable, a literal value, or a complex expression constructed with the [Expression Editor Assistant](#). For example, an assignment step might copy the value "John" to the contents of the variable 'FirstName'. Once you declare a variable, it is available to all steps throughout the handler.

Note: Once you have declared a variable in a handler, you cannot delete that variable. It will still be available throughout the handler.

Statement Page

From the Statement page you can declare a new variable and assign it a value, or select an existing variable and assign a value to it.

Variable

The name of the variable you are creating or assigning a value to. You can select new from the drop-down list to declare a new variable. The variables that appear in this list are variables that you have already declared in this handler or subroutine.

Value

Type a literal value for the variable, or create a complex expression using the Expression Editor Assistant. You can only assign values that are legal for the type of variable you selected in the previous step. For example, if the variable type is String, then the value must be string in quotes.

Exit Paths

Next

This step always exits along the Next exit path.

Related Topics

[Data types](#)

[Expression Editor Assistant](#)

Condition Tool

This Basic tool tests a conditional expression and follows the link associated with the result of that test. A conditional expression resolves a Boolean value (either True or False). Therefore, a conditional step has a True exit and a False exit.

For example, a conditional step might compare a variable 'FirstName' to a string value "John." If 'FirstName' contains the word "John", then the handler continues execution on the True branch of the conditional step. However, if 'FirstName' does not contain the word "John", then the handler continues on the False branch.

Statement Page

On the statement page you can view a conditional expression you have built, or start the Expression Editor Assistant to create or edit a condition.

Current Condition

This parameter contains the conditional expression. You can also type a new condition, or build a complex expression using the Expression Editor Assistant. Remember that the expression you build must result in a Boolean value.

If you encounter a variable in the Current Condition parameter, it is always a Boolean variable that is assigned a value somewhere else in the handler, or was created in the initiator step that started the handler or subroutine. Likewise, you can evaluate the value of a Boolean variable whose value was assigned elsewhere in a handler or subroutine.

It is important to keep track of the logic being used in setting up the conditional expression, especially when it is the condition of a Boolean variable that is being checked. For example, consider a Boolean variable 'CallReceived' that has the value of "False". If the condition expression is written simply to look at the variable 'CallReceived' without an associated value given, then the handler will continue on the False branch because the value of 'CallReceived' is "False". On the other hand, if the expression compares the variable 'CallReceived' to the value "False", then the handler will continue execution on the True branch because it is true that the value is "False".

Exit Paths

True

If the condition evaluated by this step has a value of true, this step takes the True exit path.

False

If the condition evaluated by this step has a value of false, this step takes the False exit path.

Related Topics

[Boolean values](#)

[Expression Editor Assistant](#)

Notify Debugger

This Basic tool creates a breakpoint in a handler or subroutine you are debugging. In debug mode, a handler will stop executing at the Notify Debugger step. If you have more than one Notify Debugger step in a handler or subroutine, you will be prompted when you start a debug session to select which Notify Debugger step to use as the first breakpoint.

If there is at least one Notify Debugger step in a handler, then the Select Debug Start Point(s) dialog will be displayed which will list all Notify Debugger steps as well as the Initiator. If there are no Notify Debugger steps in a handler, the initiator will automatically be set as the first breakpoint when a user begins a debugging session.

Settings Page (Notify Debugger)

Label

The unique label for this debug session. If you add another Notify Debugger step to this handler, it must have different name.

Exit Paths

Next

When this step has finished, control passes to the step you have linked to this step. If there are no links from this exit, the handler or subroutine that contains this step ends.

Related Topics

[Debugging](#)

[Set a breakpoint](#)

Selection Tool

This Basic tool tests a set of conditions. The tool begins testing with the first condition in the list, then the second, and so on. Each condition in the list has a corresponding exit path. The first condition that evaluates to true causes the handler to follow the exit path associated with that condition. For example, if the value of 'FirstName' is "John", go to the next step called "Hello John." If the value of 'FirstName' is "Jane" then go to the next step called "Hello Jane." If no conditions evaluate to True, the Default exit path is taken.

Statements Page

On the Condition Page you can add, edit, delete, or change the order of the conditions. The conditions in the list are evaluated in the order they are listed. The first true condition's exit path will be followed.

Condition list

This is a list of any conditions you have built using the Add button or Edit button. There is an exit path for every condition in this step.

Add button

Click this button to create a new condition for the Condition list using the [Expression Editor Assistant](#). Adding a condition will also add an exit path that corresponds to that condition.

Edit button

Click this button to edit a condition you selected from the Condition list.

Delete button

Click this button to delete a condition you selected from the Condition list. Deleting a condition will also delete its corresponding exit path.

Move Up button

Click this button to move a condition you selected up in the Condition list.

Move Down button

Click this button to move a condition you selected down in the Condition list.

Exit Paths

The exit paths for this tool are the conditions you type in the Condition list. An exit path will appear for each condition you type. A default exit path is always present, and this default exit path is taken if no conditions evaluate true.

Related Topics

[Expression Editor Assistant](#)

Table Lookup

This Basic tool retrieves data from a table. These tables are created and maintained in Interaction Administrator. See the Table Editor topics in the Interaction Administrator online help for more information on tables.

This tool allows you to query specific columns within a table for specific values, and then return values from the matching row. Table lookups are case sensitive. In the example table below, you could perform a lookup on the LastName column where LastName = Adams, and then return the email address associated with Adams. You can bind the returned value to a variable that can be used within the handler.

Any columns you want to perform lookups against must be indexed. When you create the table in Interaction Administrator, you can determine which columns are indexed, and the type of index to apply. Tables support two types of indexing: unique and multiple

Unique – each entry in the column is unique (e.g., account number) amongst all the entries in that column. Table Editor warns you if it finds duplicates. Lookups on unique index columns are faster than on columns with duplicate values. In the example table below, LastName has a Unique index.

Since only zero or one rows will be returned from a lookup on a unique column, the type of variables to which you bind the returned row values will be string. See Column Bindings below for more information.

Multiple – each entry in the column may have one or more occurrences of that value in the column (e.g., account type, date, etc.). Lookups on multiple value indexes are slower than on columns with unique indexes. In the example table below, FirstName has a Multiple index.

Since zero, one, or more rows will be returned from a lookup on a multiple column, the type of variables to which you bind the returned values is a list of strings. See Column Bindings below for more information.

Example Table:

LastName	FirstName	EmailAddress
Adams	Jane	JaneA@foo.com
Birch	Steve	SteveB@foo.com
Conner	Jane	JaneC@foo.com

In the table above, you could search the LastName column for Adams, and then return the email address associated with Adams.

Settings

Table Name

The name of the table as configured in Interaction Administrator. Any tables you have created in Interaction Administrator are listed here.

Lookup Expressions

Configures the column(s) to search and the value(s) you want to search for. Click the Add button to add a column against which to do the lookup and an expression for which to search in that column. Click the Modify button to modify the selected expression. Click the Delete button to remove the selected lookup column and its expression from the list.

From the example table above, you would specify the following to find the row where LastName = "Adams":

Column	Expression
LastName	"Adams"

From the example table above, you would specify the following to find the row where LastName = "Adams" and FirstName = "Jane":

Column	Expression
LastName	"Adams"
FirstName	"Jane"

Note: CIC inserts an AND condition between multiple lookup expressions when this tool executes. There is no way to specify an "or" or any other condition between lookup expressions on a table. If you desire more robust conditions, we recommend that you use the database tools and an external database.

Column Bindings

The value(s) to extract from the row(s) returned from the lookup. You may specify more than one value to return.

From the example table above, if you wanted to return the first name and email address for the row in which LastName = "Adams," you would use the following Column Bindings:

Column	Variable
FirstName	StrFirstName
EmailAddress	StrEmailAddress

When this step executes, it populates StrFirstName with "Jane" and StrEmailAddress with "JaneA@foo.com."

In another example from the example table above, if you wanted to search for the last name and email address for the row in which FirstName = "Jane", you would use the following Column Bindings.

Note that in this example, you are searching on a multiple index column (where more than one row can have the value of Jane, so you must specify a List of String variable to hold the output values):

Column	Variable
LastName	ListOfStrLastName
EmailAddress	ListOfStrEmailAddress

When this step executes, it populates the ListOfStrLastName with two elements: "Adams" and "Conner." It populates the parallel ListOfStrEmailAddress variable with two values: "JaneA@foo.com" and "JaneC@foo.com." (where "parallel list" means that the 1st element in one list correlates to the first element in a second list, and so on, until the nth element in the first list correlates to the nth element in the second list).

Exit Paths

Success

If the value was found and returned, this tool takes the Success exit path.

Table Not Found

This is returned if the table you requested data on is not found. This can occur if the table has been removed or renamed since this handler was published.

Sync Error

The Sync Error path is taken if someone has changed an index from Unique to Multiple and the variable types the handler passed in are incorrect according to the index type. For example, the handler author selected a unique index column, bound some variables, then published that handler. But then someone went into Table Editor and changed that index column from unique to non-unique. Then the handler executed. The Table Lookup tool will return Sync Error (because the bound variables were of type String instead of List of String), and the bound strings would receive the values of the first row found.

Failure

The tool has failed for some other reason, such as an exception occurred, or the server is not responding, or your input values are incorrect.

Value not Found

This tool takes the Value not Found exit path if the value you are searching for in your Lookup Expression or Column Bindings is not found.

Write Trace Message

This Basic tool writes a message to an CIC log file (ip.log) along with the handler name, timestamp, threadID, and step ID. In order to see the user defined trace messages created by this tool, tracing must be turned up using the command:

```
ICTraceA -set ip all all
```

The IP log file is stored under \\3\ic\logs\... by default.

Inputs

Trace Message

The string to write to the handler log.

Trace Level

The value representing the trace level.

- 1-20 = Error
- 21-40 = Warning
- 41-100 = Note

Exit Paths

Next

This tool always takes the Next exit path.

Buffer

Introduction to Buffer Tools

Click on one of the tools below for more information about that tool.

[BufferGetDate](#)

[BufferGetInteger](#)

[BufferGetLength](#)

[BufferGetString](#)

[BufferHandleToInteger](#)

[BufferPutDate](#)

[BufferPutInteger](#)

[BufferPutString](#)

[CopyBuffer](#)

[CreateBuffer](#)

[DecodeBuffer](#)

[DeleteBuffer](#)

[EncodeBuffer](#)

[IntegerToBufferHandle](#)

BufferGetDate

This Buffer tool gets a date from the binary buffer.

Inputs

Buffer Handle

The buffer handle that contains the date.

Offset

An integer of the offset in the buffer. Offsets are in bytes and 0 based.

Date Format

The format to use to add the date.

Format can be:

- **ISO8601**: This selection will use the ISO 8601 format in the local time zone, e.g.: 20060801140725
- **ISO9801 UTC**: This selection will use the ISO 8601 format in the UTC time zone, e.g.: 20060801130725Z
- **Custom**: This selection will use a Custom Format string. This format uses the standard C function strftime (<http://msdn.microsoft.com/en-us/library/fe06s4ak.aspx>)

Custom Format

The custom date format to use. This field is only available if you select Custom in the Date Format field. The [Expression Editor Assistant](#) displays combinations of available categories and symbols and values.

Encoding

This parameter selects the custom encoding to be used.

- UCS-2
- UTF-8
- EBCDIC

Outputs

Date

This datetime value contains the extracted DateTime.

Updated Offset

This integer contains the original offset, updated by the size of what was read. Offsets are in bytes and 0 based.

Exit Path

Success

This step takes this path on success.

Invalid Handle

This step takes this path if the buffer handle was invalid

Invalid Codepage

This step takes this path if the EBCDIC code page is not supported.

Invalid Offset

This step takes this path if the offset did not point inside the buffer.

Invalid Format

This step takes this path if date format is invalid.

BufferGetInteger

This Buffer tool retrieves an integer from the binary buffer.

Inputs

Buffer Handle

The buffer handle that contains the integer.

Offset

The integer value for the offset in the buffer. Offsets are in bytes and 0 based.

Encoding

How we encode the integer. The endianness is the byte ordering in memory used to represent some kind of data.

- Little Endian
- Big Endian

Bits

The number of bits to store the integer. If the integer is greater than the storage, it is simply truncated. E.g. 0x1234 stored on 8 bits will give 0x34.

The values are 8, 16, 32.

Outputs

Integer

The integer value from the buffer.

Updated Offset

This integer contains the original offset, updated by the size of what was read. Offsets are in bytes and 0 based.

Exit Paths

Success

This step takes this path on success.

Invalid Handle

The tool takes this path, if the buffer handles was invalid.

Invalid Offset

This step takes this path if the offset did not point inside the buffer.

BufferGetLength

This Buffer tool returns the size (in bytes) of the data contained in a binary buffer.

Inputs

Buffer Handle

The buffer handle of the buffer you want to measure.

Outputs

Length

An integer value of the size of the buffer.

Exit Paths

Success

This step takes this path on success.

Invalid Handle

The tool takes this path, if the buffer handles was invalid.

BufferGetString

This Buffer tool retrieves an integer from the binary buffer.

Inputs

Buffer Handle

The buffer handle that contains the integer.

Offset

The integer value for the offset in the buffer. Offsets are in bytes and 0 based.

Read Mode

This selection is how the length is stored. It can be:

- Prepended Length
- Read until a zero
- Custom

Length

If the Read Mode is "Custom", this integer value is the number of characters to read.

Encoding

This parameter selects the custom encoding to be used.

- UCS-2
- UTF-8
- EBCDIC

Codepage

This integer is the EBCDIC code page to use. This field is only available if the EBCDIC custom encoding is selected. Currently, only the code page 290 is supported.

Outputs

String

The string value from the buffer.

Updated Offset

This integer contains the original offset, updated by the size of what was read. Offsets are in bytes and 0 based.

Exit Paths

Success

This step takes this path on success.

Invalid Handle

The tool takes this path, if the buffer handles was invalid.

Invalid Codepage

The tool takes this path if the codepage value specified is not valid.

Invalid Offset

This step takes this path if the offset did not point inside the buffer.

BufferHandleToInteger

This Buffer tool converts the unique identifier of a buffer handle to an integer. Once it is an integer the value can be sent from a handler to another one. Once in another handler, the integer can be used to fetch the buffer back as long as it has been kept alive.

Inputs

Buffer Handle

The buffer handler you want to convert to an integer.

Outputs

Value

An integer value of the buffer handle.

Exit Paths

Success

This step always takes this path.

BufferPutDate

This Buffer tool appends a date at the end of the given binary buffer.

Inputs

Buffer Handle

The buffer handle that contains the date.

Date

The date value to add to the buffer.

Date Format

The date format to use:

- ISO 8601
- ISO 8601 UTC
- Custom

Custom Format

The custom date format to use. This field is only available if Custom is selected in the Date Format field. The Expression Editor Assistant appears to display combinations of available categories and symbols and values.

Encoding

This parameter selects the custom encoding to be used.

- UCS-2
- UTF-8
- EBCDIC

Codepage

An expression selected from one of the various categories in the Expression Editor Assistant. This field is only available if the EBCDIC custom encoding is selected. Currently, only the code page 290 is supported.

Exit Paths

Success

This step takes this path on success.

Invalid Handle

The tool takes this path if the buffer handle was invalid.

Invalid Codepage

The tool takes this path if codepage value that was specified is not supported.

Invalid Format

The tool takes this path if the custom format specified is not supported.

BufferPutInteger

This Buffer tool appends an integer at the end of the given binary buffer.

Inputs

Buffer Handle

The buffer handle that is the target.

Integer

The integer value to add to the end of the buffer.

Encoding

How we encode the integer. The endianness is the byte ordering in memory used to represent some kind of data.

- Little Endian
- Big Endian

Number of bits

The number of bits to store the integer. If the integer is greater than the storage, it is simply truncated. For example, 0x1234 stored on 8 bits will give 0x34.

The values are 8, 16, 32.

Exit Paths

Success

This step takes this path on success.

Invalid Handle

The tool takes this path if the buffer handle was invalid.

BufferPutString

This Buffer tool appends a string to the end of the given binary buffer.

Inputs

Buffer Handle

The buffer handle that contains the integer.

String

The string value to add to the buffer.

Encoding

This parameter selects the custom encoding to be used.

- UCS-2
- UTF-8
- EBCDIC

Codepage

This integer is the EBCDIC code page to use. This field is only available if the EBCDIC custom encoding is selected. Currently, only the code page 290 is supported.

Prepend Size

If checked, the size in characters of the string will be added to the buffer before the string itself.

Note: The size is stored in the Network format (i.e. in Big Endian)

Append Zero terminator

If checked, an additional null character is added to the end of the buffer.

Exit Paths

Success

This step takes this path on success.

Invalid Handle

The tool takes this path, if the buffer handle was invalid.

Invalid Codepage

The tool takes this path if the Codepage value specified is not supported.

CopyBuffer

This Buffer tool copies a given binary buffer into another binary buffer.

Note: The target buffer is entirely overwritten. Any existing data in the target buffer will be lost.

Inputs

Source Buffer Handle

The buffer handler you want to copy.

Target Buffer Handle

The buffer handle of the target buffer.

Exit Paths

Success

This step takes this path on success.

Invalid Handle

The tool takes this path, if one of the buffer handles was invalid.

CreateBuffer

This Buffer tool creates a new empty binary buffer.

Outputs

Buffer Handle

This is a Buffer Handle used to reference the buffer in other tools.

Exit Paths

Success

This step always takes the Success path.

DecodeBuffer

This Buffer tool de-serializes a string into a binary buffer. The string must be in the format "0x1234".

Note: Any existing data in the buffer handler will be replaced by the new data.

Inputs

Buffer Handle

The buffer handle to be overwritten.

String

The string value that contains the serialized buffer.

Exit Paths

Success

This step takes this path on success.

Invalid Handle

The tool takes this path, if the buffer handles was invalid.

DeleteBuffer

This Buffer tool destroys a given binary buffer.

Inputs

Buffer Handle

This is the Buffer Handle of the buffer handle to destroy.

Exit Paths

Success

This path is taken if the buffer is destroyed.

Invalid Handle

This path is taken if the buffer handle was invalid.

EncodeBuffer

This Buffer tool serializes a binary buffer into a string. The resulting string is in the format "0x1234".

Inputs

Buffer Handle

The buffer handle to be serialized.

Outputs

String

The string value that contains the serialized buffer.

Exit Paths

Success

This step takes this path on success.

Invalid Handle

The tool takes this path, if the buffer handles was invalid.

IntegerToBufferHandle

This Buffer tool converts an integer to a buffer handle.

Inputs

Value

The integer value that points to a buffer.

Outputs

Buffer Handle

The buffer handle of the new buffer created.

Exit Paths

Success

This step takes this path on success.

Invalid Handle

The tool takes this path, if the buffer handles was invalid.

Calendar

Introduction to Calendar Tools

The Calendar Tools can be used in a handler to place, retrieve or edit data on a calendar server. Presently, CIC only supports the iPlanet, MS Exchange and Outlook calendar servers. This release only supports Events, and only the Start Time, End Time, Location, Description, and Summary Properties. Other calendars and features will be added in future releases.

The calendaring system is set up automatically when CIC is installed to coincide with the mail server that is selected during setup.

Click on one of the tools below for more information about that tool.

[Add Event](#)

[Add Task](#)

[Are Available](#)

[Delete Event](#)

[Delete Task](#)

[Edit Event](#)

[Edit Task](#)

[Find Next](#)

[Get Event](#)

[Get Events](#)

[Get Task](#)

[Get Tasks](#)

[Is Available](#)

[Logon](#)

[Open Calendar](#)

[Search Events](#)

[Search Events Ex](#)

[Search Tasks](#)

[Search Tasks Ex](#)

Add Event

This Calendar tool adds an event to the specified calendar.

Inputs

Calendar ID

This is a Calendar ID from the Open Calendar tool.

Start Date

The start of the event.

End Date

The end of the event. This time must be later than the defined Start Time.

Location (optional)

The location of the event.

Description (optional)

The description of the event.

Summary (optional)

A summary of the event.

Output

Event ID

An ID which can be used to later retrieve the event.

Exit Paths

Success

This path is taken if the event was added successfully.

Failure

This path is taken if the operation fails.

Add Task

This Calendar tool adds a task to a specified calendar.

Inputs

Calendar ID

This is a Calendar ID from the [Open Calendar](#) tool.

Due Date

Date/time string indicating the date the task is due.

Location

The location of the task.

Description

String containing a description of the task.

Summary

String summarizing the task.

Status

The current status of the task.

Outputs

Task ID

An integer that represents the task and is used as input to other tools.

Exit Paths

Success

This path is taken if the task is successfully added to the calendar.

Failure

This path is taken if the operation fails.

Are Available

This Calendar tool checks the availability of the given time slot across multiple calendars.

Inputs

Session ID

This is a Session ID from the Logon tool.

Calendar Names

A list of strings containing the names of all the calendars to be checked for availability.

See [List Tools](#) for information on creating lists of strings.

Start Date

The start of the time block being checked for availability.

End Date

The end of the time block being checked for availability. This date must later than the date specified in the Start Date field.

Exit Paths

Available

This path is taken if the given time slot is available on all of the given calendars.

Not Available

This path is taken if the given time slot is not available on all of the given calendars.

Failure

This path is taken if the operation fails. The most common reasons for failure are invalid username, invalid password, or invalid host. Other unexpected errors could also cause failure.

Delete Event

This Calendar tool deletes an event from a specified calendar.

Inputs

Calendar ID

This is a Calendar ID from the [Open Calendar](#) tool.

Event ID

The ID of an event such as the [Add Event](#) tool.

Exit Paths

Success

This path is taken if the event is successfully deleted from the specified calendar.

Failure

This path is taken if the operation fails.

Delete Task

This Calendar tool deletes a specified task from a specified calendar.

Inputs

Calendar ID

This is a Calendar ID from the [Open Calendar](#) tool.

Task ID

This is a Task ID from the [Add Task](#) tool.

Exit Paths

Success

This path is taken if the task is successfully deleted.

Failure

This path is taken if the operation fails.

Edit Event

This Calendar tool edits an event to a specified calendar.

Note: If Edit Event is used to edit an entry created outside of the tools, any properties besides the Start Date, End Date, Location, Description, and Summary will be deleted from the updated entry.

Inputs

Calendar ID

This is a Calendar ID from the Open Calendar tool.

Event ID

The ID of an event such as the Add Event tool.

Start Date (optional)

The new start date.

End Date (optional)

The new end date. This time must be later than the defined Start Time for the event.

Location (optional)

The new location.

Description (optional)

The new description.

Summary (optional)

The new summary.

Exit Paths

Success

This path is taken if the event was updated successfully.

Failure

This path is taken if the operation fails.

Edit Task

This Calendar tool updates an existing task.

Inputs

Calendar ID

This is a Calendar ID from the [Open Calendar](#) tool.

Task ID

This is a Task ID from the [Add Task](#) tool.

Due Date

Date/time string indicating the date the task is due.

Location

The location of the task.

Description

String containing a description of the task.

Summary

String summarizing the task.

Status

The current status of the task.

Exit Paths

Success

This path is taken if the task is successfully updated.

Failure

This path is taken if the operation fails.

Find Next

This Calendar tool finds the next available slot in the calendars for a given block of time.

Note: The duration is the minimum amount of time to search for. It is possible to request a 30 minute block of time, but have a 60 minute block of time returned.

Inputs

Session ID

This is a Session ID from the Logon tool.

Calendar Names

A list of Calendar Names.

Start Date

The earliest time to start.

End Date

The latest time to end. This time must be later than the defined Start Time.

Duration

Integer - The minimum amount of time in minutes.

Output

Start Date

The start of an available block.

End Date

The end of an available block.

Exit Paths

Available

This path is taken if an available time slot is successfully returned.

Not Available

This path is taken if there is no available time slot within the specified time period.

Failure

This path is taken if the operation fails.

Get Event

This Calendar tool retrieves an event from a specified calendar.

Inputs

Calendar ID

This is a Calendar ID from the Open Calendar tool.

Event ID

The ID of an event (from the Add Event tool for example).

Outputs

Start Date

The start date of the event.

End Date

The end date of the event. This date must be later than the defined Start Date.

Location

The location of the event.

Description

String - The description of the event.

Summary

The summary of the event.

Organizer

A string containing the e-mail addresses of the event organizer.

Organizer CN

A string containing the friendly name or external e-mail of the organizer. The calendar provider may not provide this value, so the string may be blank.

Attendees

A string of strings containing the email addresses of the event attendees.

Attendees CNs

A list of strings containing the friendly name or external email of the attendees. The calendar provider may not provide this value, so the string may be blank.

Exit Paths

Success

This path is taken if the event was retrieved successfully.

Failure

This path is taken if the operation fails.

Get Events

This Calendar tool gets a list of all events in a given range.

Inputs

Calendar ID

This is a Calendar ID from the Open Calendar tool.

Start Date

The start of the range.

End Date

The end of the range. This date must be later than the defined Start Date.

Output

Event IDs

A list of Event IDs.

Exit Paths

Success

This path is taken if the list of events is successfully retrieved.

Failure

This path is taken if the operation fails.

Get Task

This Calendar tool retrieves information on a single task from a calendar.

Inputs

Calendar ID

This is a Calendar ID from the [Open Calendar](#) tool.

Task ID

This is a Task ID from the [Add Task](#) tool.

Outputs

Due Date

Date/time string indicating the date the task is due.

Location

The location of the task.

Description

String containing a description of the task.

Summary

String summarizing the task.

Status

The current status of the task.

Exit Paths

Success

This path is taken if the tasks are successfully retrieved from the calendar.

Failure

This path is taken if the operation fails.

Get Tasks

This Calendar tool retrieves all tasks within a specified span of dates.

Inputs

Calendar ID

This is a Calendar ID from the [Open Calendar](#) tool.

Start Date

The start date of the tasks.

End Date

The end date of the tasks. This time must be later than the defined Start Time for the tasks.

Outputs

Task IDs

This list of IDs corresponding to the tasks that fall within the specified start and end dates.

Exit Paths

Success

This path is taken if the tasks are successfully retrieved from the calendar.

Failure

This path is taken if the operation fails.

Is Available

This Calendar tool checks a given time slot on the calendar for availability.

Inputs

Calendar ID

This is a Calendar ID from [Open Calendar](#) tool.

Start Date

The start of the time slot.

End Date

The end of the time slot. This date must be later than the defined Start Date.

Exit Paths

Available

This path is taken if the specified time slot is available.

Not Available

This path is taken if the specified time slot is not available.

Failure

This path is taken if the operation fails.

Logon

This Calendar tool establishes a session with the calendar server.

Inputs

Username

A string representing a username on the calendar server. Note that this field is case sensitive.

NOTE: For MS Exchange users, this field also needs to include the domain name if the calendar server is on a different domain from the CIC server. The format for this should be "domain_name\user_name".

Password

A string representing the password for the given user.

Host

A string representing the host (and possibly port) of the calendar server.

Note for MS Exchange Users: the format for this string should be, "http://your_OutlookWebAccess_servername/exchange/User_Name"

Note for iPlanet Users: the format for this string should be, "[iPlanet_Servername:port#](#)". For example, "[iPlanet:81](#)".

Output

Session ID

An integer that represents an established session and is used as input to other tools.

Exit Paths

Success

This path is taken if the session is successfully established.

Failure

This path is taken if the operation fails. The most common reasons for failure are invalid username, invalid password, or invalid host. Other unexpected errors could also cause failure.

Open Calendar

This Calendar tool opens a calendar.

Inputs

Session ID

An integer representing the Session ID from the Logon tool.

Calendar Name

A string representing the name of the calendar to open. This field is case sensitive.

Note for MS Exchange Users: This field should be formatted as follows:

"http://your_OutlookWebAccess_servername/exchange/User_SMTP_address"

Note for iPlanet Users: this field should be formatted as follows:

"Calendar_name"

Output

Calendar ID

An integer that represents the open calendar and is used as input to other tools.

Exit Paths

Success

This path is taken if the calendar is successfully opened.

Failure

This path is taken if the operation fails. The most common reasons for failure are invalid username, invalid password, or invalid host. Other unexpected errors could also cause failure.

Search Events

This Calendar tool retrieves all calendar events that match the specified parameters.

Inputs

Calendar ID

This is a Calendar ID from the [Open Calendar](#) tool.

Start Date

The start of the event.

End Date

The end of the event. This time must be later than the defined Start Time.

Location

The location of the event.

Description

The description of the event.

Summary

A summary of the event.

Outputs

Event IDs

The list of IDs corresponding to the events retrieved by the search.

Exit Paths

Success

This path is taken if the Event ID list is successfully retrieved.

Failure

This path is taken if the operation fails.

Search Events Ex

This Calendar tool finds all events in a calendar that fall within a specified date range and that match an input search expression.

Inputs

Calendar ID

This is a Calendar ID from the [Open Calendar](#) tool.

Start Date

The start of the event.

End Date

The end of the event. This time must be later than the defined Start Time.

Search Expression

The search expression used by this tool should use the following format:

```
search_expr ::= search_expr2
             | search_expr2 OR search_expr
search_expr2 ::= search_expr3
             | search_expr3 AND search_expr2
search_expr3 ::= search_term
             | NOT search_term
search_term  ::= value_expr relop value_expr
             | value_expr LIKE string_constant
             | value_expr NOT LIKE string_constant
             | value_expr IS NULL
             | value_expr IS NOT NULL
             | "(" search_expr ")"
relop       ::= "=" | ">" | "<" | ">=" | "<=" | "<>"
value_expr  ::= property_name
             | constant
constant    ::= string_constant
             | int_constant
property_name ::= ALPHA (ALPHA | DIGIT | "-")*
string_constant ::= "'" (<any character except "'"> | "'')* "'"
int_constant  ::= <C-style decimal or hexadecimal constant>
```

In a LIKE expression, use a "_" as a wildcard to match a single character and a '%' to match matches any number of characters.

Outputs

Event IDs

The list of IDs corresponding to the events retrieved by the search.

Exit Paths

Success

This path is taken if the Event ID list is successfully retrieved.

Failure

This path is taken if the operation fails.

Search Tasks

This Calendar tool retrieves all calendar tasks that match the specified parameters.

Calendar ID

This is a Calendar ID from the [Open Calendar](#) tool.

Start Date

Start of the date range for which to retrieve the tasks that are due.

End Date

End of the date range for which to retrieve the tasks that are due.

Location

The location of the task.

Description

String containing a description of the task.

Summary

String summarizing the task.

Status

The current status of the task.

Outputs

Task IDs

The list of IDs corresponding to the tasks retrieved by the search.

Exit Paths

Success

This path is taken if the Task ID list is successfully retrieved.

Failure

This path is taken if the operation fails.

Search Tasks Ex

This Calendar tool retrieves all calendar tasks that are scheduled for a specified period of time and/or that meet a specified description.

Calendar ID

This is a Calendar ID from the [Open Calendar](#) tool.

Start Date

Date/time string indicating the beginning of the date range.

End Date

Date/time string indicating the end of the date range.

Search Expression

Specific expression to search for with the task body. See [Search Events Ex](#) for details on formatting this search expression.

Outputs

Task IDs

The list of IDs corresponding to the tasks retrieved by the search.

Exit Paths

Success

This path is taken if the Task ID list is successfully retrieved.

Failure

This path is taken if the operation fails.

Database

Database Tools

The Database tools allow handlers and subroutines to read from and update databases. In the default handlers that ship with CIC, many of the reporting and statistic gathering handlers use database tools, and the CustomCallDisconnectMonitor handler writes call recording information to a database for use by the Interaction Recorder application. Database tools have many uses because they allow you to read from and write to a database outside of the CIC server. While their uses are virtually unlimited, some other uses are looking up passwords and retrieving information to display in web pages.

Typically, the Database tools are used in the following order within a handler, as illustrated by this diagram, entitled [The order in which Database tools should be used](#). Use this multi-step process as guide when creating database functionality within a handler.

Note: In rare circumstances where you have a very large number of tables to load and/or your server is fairly busy at the time you need to load the tables, it is possible that the system might timeout before the tables have finished loading. If this happens, the timeout period can be reset to whatever you desire by entering the following command in a DOS window:

```
/DBQueryTimeout=X
```

Where X is an integer representing the number of seconds to wait before timing out.

Step One: DB Open to open an ODBC data source

With [DB Open](#) you specify the data source, username, password, name of the variable to hold the connection handle, and the maximum number of connections. Most other Database tools use this connection.

Step Two: DB Get Connection to a connection with the database

[DB Get Connection](#) attempts to open the actual connection. Once the connection is established, DB Query, DB Fetch, and DB SQL Exec can perform operations on the database.

Step Three: DB Query can find records. DB SQL Exec can perform operations on the database.

With a connection established, [DB Query](#) can execute a Where clause to create a result set, preparing the records for a DB Fetch step. Each connection can only store one result set, so if you need to retrieve multiple result sets for multiple fetches, you should use additional DB Get Connection steps. [DB SQL Exec](#) can insert, update, or delete records, or run a script stored on the database server.

Step Four: DB Fetch accesses the result set.

[DB Fetch](#) retrieves a record(s) from the result set. The values retrieved are stored in the variables set up in the DB Query step.

Step Five: DB Release Connection

If the handler no longer needs to access the database, use a [DB Release Connection](#) to drop the connection to the database. If you opened multiple connections, you can use this step to close any connections that you do not need.

Step Six: DB Close

When the handler finishes running, you should close the ODBC data source with a [DB Close](#) step.

Database tools:

[DB Close](#)

[DB Fetch](#)

[DB Flush](#)

[DB Get Connection](#)

[DB Get Data List](#)

[DB Open](#)

[DB Put Data List](#)

[DB Query](#)

[DB Release Connection](#)

[DB SQL Exec](#)

[DB SQL Exec2](#)

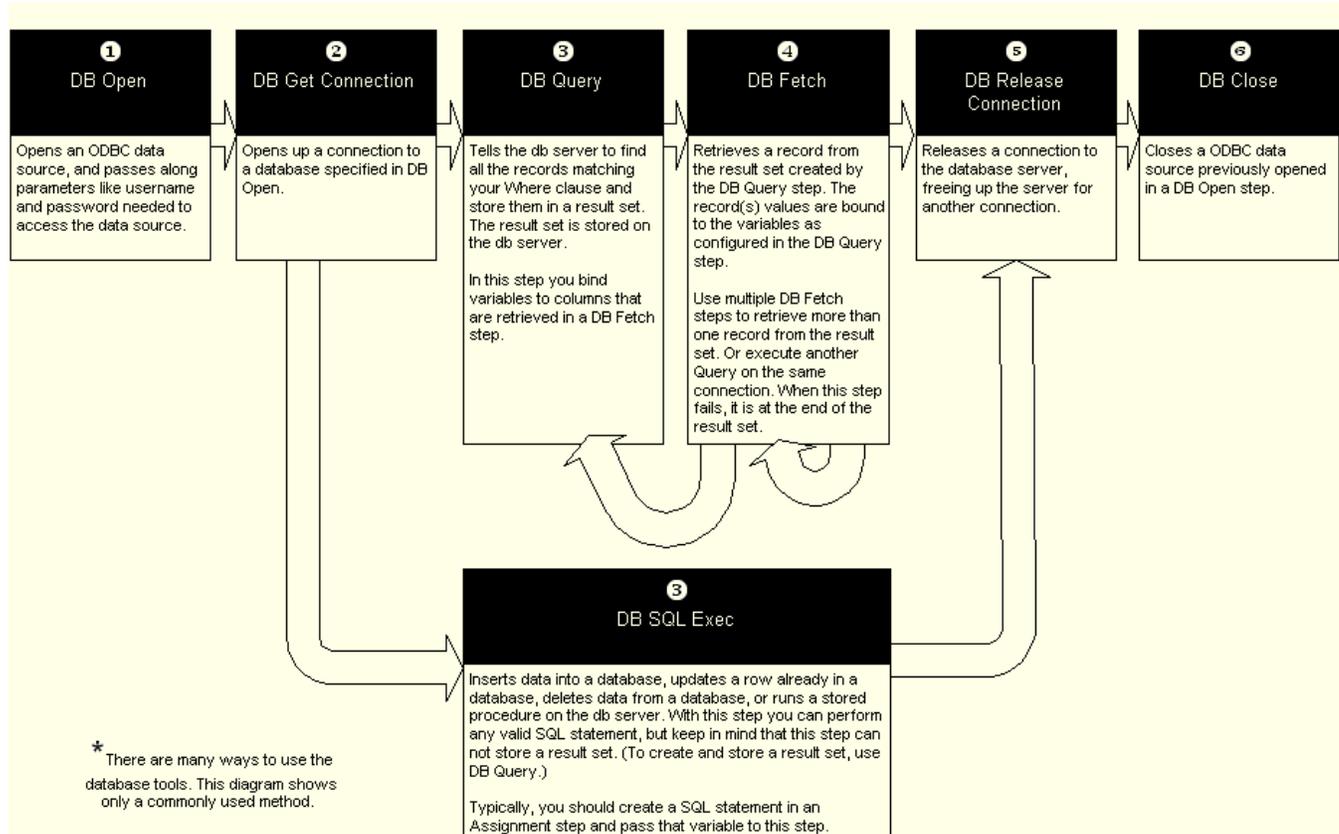
[DB Stored Procedure](#)

Related Topics:

[Supported Databases](#)

The Order in Which Database Tools Should Be Used

This diagram suggests an order in which the [Database Tools](#) should be used. For best results when printing, set your paper orientation to Landscape.



DB Stored Procedure Definitions

Displays and allows you to edit the definition of a stored procedure. This is either the definition as passed in by the ODBC driver, or the definition that you have modified previously in this dialog box.

Use this dialog to augment a stored procedure definition if the ODBC driver did not pass in a complete definition. ODBC drivers differ in the amount of information that they pass to Interaction Designer. Some will pass all input and output parameters, some won't. Use this dialog if you realize that your ODBC driver is passing an incomplete definition.

Note: The definition that you create here is stored on the CIC server. When you select this definition in a DB Stored Procedure step and then save that handler document, the definition will also be stored within the handler. It is possible that the definition stored on the CIC server will become different from the definition stored with the handler. This can occur if you make a change to the stored procedure itself on the database server, and then only update the copy on the CIC server or the copy in the handler. If you open a handler and Interaction Designer detects that there is a difference in the definition in the handler and the definition on the CIC server, you will be presented with a dialog asking you to select the current definition (the one you want to keep).

Note: Refer to the release notes for last minute notes and cautions for using this tool.

DSN (Catalog =)

The name of the data source on which the procedure was created. The concept of the catalog varies from one RDBMS vendor to the next. It usually corresponds to the physical database, but there are exceptions. For example, with dBase & FoxPro it usually corresponds to the directory in which the data files reside. While some RDBMSs allow you to dynamically switch catalog contexts within a connection, CIC currently doesn't support this, and assumes the catalog is tied to the ODBC DSN. So, if access to multiple catalogs is required, you must create and configure multiple ODBC DSNs.

Schema

The schema in which the stored procedure resides. The "schema" is the official ANSI term for what used to be commonly known as the "qualifier". It is usually the creator/owner of the database object. Worth noting is that Sybase and MS SQL Server have a special schema name that is used when the owner of the object is also the *database owner*: "dbo".

Stored Procedure

The name of the stored procedure as it is stored on the DB server. You cannot change the name of the stored procedure from this field.

Parameters

After you have specified a stored procedure in the previous parameter, this list is populated with any parameters passed in by the ODBC driver, and any parameters that you have previously added to this definition. Add or edit parameters in this list as necessary. Adding or editing a parameter does not change the actual stored procedure (on the database server), only Interaction Designer's definition of it. Any change to the actual procedure must be performed on the database server where the stored procedure resides.

Result Columns

After you have specified a stored procedure, this list is populated with any result columns passed in by the ODBC driver, and any result columns that you have previously added to this definition. Add or edit result columns in this list. Adding or editing a result column does not change the actual stored procedure, only Interaction Designer's definition. Any change to the actual procedure must be performed on the database server where the stored procedure resides.

Return Value Type

A value returned as a result of the processing performed by the stored procedure. The type of value returned depends on the RDBMS. For example, Oracle and DB2 allow you to return most legal SQL data types (LONG or BLOB types usually being the exceptions), but Sybase and MS SQL Server both limit return types to INTEGERS. To my knowledge, none of the RDBMSs allow complex types, like lists, to be returned.

Return Value Type Size

The amount of memory to reserve for the returned value. This field will only be enabled for SQL Types that do not have fixed size.

Return Value Type Decimal Digits

For values that are SQL Float type, this specifies the number of digits to store. This is applicable only to Decimal and Numeric types.

Related Topics

[DB Stored Procedure](#)

Supported Databases

Refer to the PureConnect Customer Care web site to view the list of supported databases for the current release of CIC.

Sample SQLExec Statements

When building SQLExec statements for use in a [DB SQL Exec](#) step, you must build the SQLExec statements in a [Assignment](#) step (using the [Expression Editor Assistant](#)) and assign it as the value of a variable. Building a SQLExec statement from the Expression Editor Assistant requires special formatting. Refer to the following examples when constructing your Where clauses:

SQLExec statement:

```
"INSERT INTO CUSTOMER_TABLE (CUSTOMER_ID, CUSTOMER_NAME, CITY, STATE) VALUES ('ACB8934', 'Joe''s Bar & Grill', 'Indianapolis', 'IN') "
```

Assign String using a literal:

```
"INSERT INTO CUSTOMER_TABLE (CUSTOMER_ID, CUSTOMER_NAME, CITY, STATE) VALUES ('ACB8934', 'Joe''s Bar & Grill', 'Indianapolis', 'IN') "
```

Assign String using variables:

```
"INSERT INTO CUSTOMER_TABLE (CUSTOMER_ID, CUSTOMER_NAME, CITY, STATE) VALUES (" & SQLStr(var_customer_id) & "," & SQLStr(var_customer_name) & "," & SQLStr(var_city) & "," & SQLStr(var_state) & ") "
```

Note: See the [SQLStr](#) help for more information on the SQLStr function.

Sample Where Clauses

When building where clauses for use in a [DB Query](#) step, you must build the Where clause in a [Assignment](#) step (using the [Expression Editor Assistant](#)) and assign it as the value of a variable. Building a Where clause from the Expression Editor Assistant requires special formatting. Refer to the following examples when constructing your Where clauses:

Where statement:

```
"SELECT CUSTOMER_ID, CUSTOMER_NAME, STATE FROM CUSTOMER_TABLE WHERE STATE='IN' "
```

Assign WhereString using a literal: "STATE='IN'"

Assign WhereString using a variable: "STATE=" & SQLStr(var_state)

Where statement:

```
" SELECT CUSTOMER_ID, CUSTOMER_NAME, STATE FROM CUSTOMER_TABLE WHERE STATE='IN' AND ACTIVE=1 "
```

Assign WhereString using a literal: "STATE='IN' AND ACTIVE=1"

Assign WhereString using variables: "STATE=" & SQLStr(var_state) & " AND ACTIVE=1"

Note: See the [SQLStr](#) help for more information on the SQLStr function.

Database Tools

DB Close

This Database tool closes an ODBC data source that was previously opened in a [DB Open](#) step.

Settings Page

On the settings page, you must specify which database variable you are going to close. Specify a database variable that you created using a [DB Open](#) step.

Database Variable

Specifies which database variable you are going to close.

Exit Paths

Success

If this step executes successfully, this step takes the Success exit path.

Failure

If this step does not execute successfully, this step takes the Failure exit path. This can occur when the database variable is invalid.

DB Fetch

This Database tool retrieves a record from a result set generated in a [DB Query](#) step.

For example, if the following result set is generated from a DB Query step.

Result Set

ID	Name	Phone	Cost
123	Bill	555-1234	\$5.00
124	Bill	555-1234	\$50.00
125	Monic a	555-9876	\$10.00
126	Bill	555-1234	\$15.00
127	Monic a	555-9876	\$5.00
Empty	Empty	Empty	Empty

The first time the DB Fetch step executes, it retrieves record with ID 123, and takes the success exit path. You can then pull values from any variables bound to the column values, as specified in the DB Query step. Then your handler would loop around and execute this DB Fetch step again. DB Fetch automatically retrieve the value of the next record, number 124. Each time the DB Fetch step executes, it retrieves the next value. When it reaches the empty record, the DB Fetch step fails and the loop ends.

In this example, you could write each value to a list of string for insertion into a web page, or some other operation.

Settings Page

Database Connection Variable

The name of a variable you created in a [DB Get Connection](#) step.

Timeout (seconds)

The number of seconds to wait for this operation to complete successfully before taking the failure path. Any value less than or equal to zero results in the default value of 60 seconds being used. You may specify a decimal value, such as 5.5.

Exit Paths

Success

If this step executes successfully, this step takes the Success exit path.

Failure

If this step does not execute successfully, this step takes the Failure exit path. This can occur when the parameters set in the DB query step were invalid, the DB connection variable is invalid, if the step times out before the fetch could be performed, or there were no more rows to be fetched from the result set. If there were no more rows to be fetched from the result set, this means either a) the fetch cycle is now complete, or b) there was no successful query performed before the fetch (this means that for this step, the handler author can't tell the difference between a "true" failure (like a lost DB connection) and a simple "you're at the end of the result set" indication.)

DB Flush

This Database tool attempts to clear all cached connections to a specific ODBC data source. This tool will only clear connections that were established in a handler (with a [DB Get Connection](#) tool), and that are inactive. The tool skips any active connections.

Settings Page

Data Source Name

The ODBC data source to which all cached connections are cleared. If you leave this parameter empty, all cached connections (established via handlers) to all ODBC data sources are cleared.

Exit Paths

Next

This tool always takes the Next exit path.

DB Get Connection

This Database tool opens up a connection to a database specified in [DB Open](#). If you are planning to use a DB Query step to retrieve a result set, keep in mind that each connection can only hold one result set. If you want to retrieve more than one result set, you should get a connection for each result set you want to retrieve. See the [DB Query](#) step for more information on result sets. This step should always be preceded by a DB Open step.

Note: You must close the current DB Connection before opening a new connection to the same DB or your results may not be reliable.

Settings Page

Database Variable

Specifies which database to open. For this parameter, enter the database variable created in a [DB Open](#) step.

Database Connection Variable

Names the database connection.

Timeout

Specifies how long this step will wait for a connection with the server before exiting Failure. The default for this value is 60 seconds. Entering a value ≤ 0 results in this step using the default value of 60 seconds. This parameter takes a real value, so you may specify a decimal, such as 2.5.

Exit Paths

Success

If this step executes successfully, this step takes the Success exit path.

Failure

If this step does not execute successfully, this step takes the Failure exit path. This can occur when the DB variable or DB connection variable is not valid, if the step times out before the connection could be performed, or the DB server's connection limit has been reached. Failures on open and connect depend on the type of database (Oracle, Sybase, etc.).

DB Get Data List

This Database tool retrieves a list of call attributes and a parallel list of values from an CIC source. The query it executes is "Select * from **Table Name Value Where Where Clause Value**". CIC Data Sources are configured in Interaction Administrator.

This tool was created for use with the Interaction Dialer application that will read and write many call attributes to and from a data source. In most cases, you can use the [Get Attribute](#) tool to retrieve the value of a default or custom call attribute.

Inputs

IC Data Source

This must be an CIC Data Source (configured in Interaction Administrator) that refers to an ODBC DSN.

Table Name

The name of the table to query. This becomes part of the search criteria when this tool executes.

Where Clause

The row to query. For example, "Agent = 4". If you specify more than one row, only the first row is used.

Attribute Name Prefix

The characters to strip from the beginning of the attribute name before placing the attribute name in the Attribute Name output list. For example, if the attribute name is "IntDialer_Attribute1" and you specify "IntDialer_" as the prefix to remove, only "Attribute1" is listed in the output list. This parameter is optional.

Attribute Name Suffix

The characters to strip from the end of the attribute name before placing the attribute name in the Attribute Name output list. For example, if the attribute name is "Attribute1_IntDialer" and you specify "_IntDialer" as the suffix to remove, only "Attribute1" is listed in the output list. This parameter is optional.

Timeout

The number of seconds to wait for the operation to complete before taking the Failure exit path. Any value less than or equal to zero results in the default value of 60 seconds being used. You may specify a decimal value, such as 5.5.

Outputs

Attribute Names

The list of strings containing the attribute names retrieved.

Attribute Values

The list of values that parallels the list of Attribute names.

Exit Paths

Success

If the operation is successful, this tool takes the Success exit path.

Failure

This tool takes the Failure exit path if the syntax of the Where clause is incorrect, if the CIC Data Source is configured incorrectly, or if the Timeout value is exceeded.

DB Open

This Database tool opens an ODBC data source and passes along parameters like username and password, which are needed to access the data source. This step is usually be followed by a [DB Get Connection](#) step.

Settings Page

Data Source

Specifies the ODBC data source you want to connect to.

User Name

Type the username set up when you created the ODBC data source. You must have a value here, even if it is not used. The user name may be entered explicitly, or you may use an expression do derive the username from variables.

Password

Type the password set up when you created the ODBC data source. You must have a value here, even if it is not used. The password may be entered explicitly, or you may use an expression do derive the password from variables.

Note: This tool does **NOT** obfuscate data entered in this parameter. This means that passwords that are entered explicitly **will** be visible to others who have access to handler files. Therefore it is recommended that literal strings not be used in any application where password security is desired. Handler authors should instead code the handler to use a lookup of a structured parameter and use the variable in this tool.

Database Variable Name

This is the name (handle) for the database you are opening.

Maximum Database Connections

The number of connections this handler will open. If you are going to be requesting multiple results from your query to the database, you can specify more connections. With one connection you can perform one query and connection and one set of fetches at a time. Multiple connections allow you to query the database, leave the result on the server, perform another query, and go back later to continue getting records.

Exit Paths

Success

If this step executes successfully, this step takes the Success exit path.

Failure

If this step does not execute successfully, this step takes the Failure exit path. This can occur when the ODBC data source is not valid, the user name or password are incorrect, the step times out before the DB can be opened, or the DB server is not available.

See Also:

[Supported Databases](#)

DB Put Data List

This Database tool writes attribute data to an ODBC CIC Data Source as a row of data. If the database table already has this row, it will be updated; otherwise, a new row will be inserted. A new insert is executed as follows: 'Insert into **Table Name Value (Comma-Separated Attribute Names)** values (**Comma-Separated Attribute Values**)'. An update is executed as follows:

'Update Table Name Value, set Attribute Name1=Attribute Value1, Attribute Name2=Attribute Value2, where Where Clause Value).

This tool was created for use with the Interaction Dialer application that will read and write many call attributes to and from a data source. In most cases, you can use the [Set Attribute](#) tool to set the value of a default or custom call attribute.

Inputs

IC Data Source

This must be an CIC Data Source (configured in Interaction Administrator) that refers to an ODBC DSN.

Table Name

The name of the table to which the attributes are written.

Where Clause

The row to write. For example, "Agent = 4". You should not specify more than one row. If you do specify more than one row, every matching row is updated, which is normally not desired behavior. This Where Clause is optional, and if omitted, a suitable Where Clause is generated automatically as long as your ODBC driver supports the *SQLPrimaryKeys* API (see note 1 below). Note that this is only optional for DB Put Data List; it is required in [DB Get Data List](#) since there is no input data to generate the clause from.

Attribute Name Prefix

The value to prepend to the attribute name before writing it to the data source. For example, if the attribute name is "Attribute1" and you specify "IntDialer_" as the prefix, "IntDialer_Attribute1" is listed in written to the data source. This parameter is optional.

Attribute Name Suffix

The value to append to the attribute name before writing it to the data source. For example, if the attribute name is "Attribute1" and you specify "_IntDialer" as the suffix, "Attribute1_IntDialer" is listed in written to the data source. This parameter is optional.

Timeout

The number of seconds to wait for this operation to complete successfully before taking the failure path. Any value less than or equal to zero results in the default value of 60 seconds being used. You may specify a decimal value, such as 5.5.

Attribute Names

A list of strings value containing the list of attribute names to be written to the data source.

Attribute Values

A list of attribute values parallel to the list of Attribute Names. If the number of values does not match the number of names, this step will take the Failure exit path.

Exit Paths

Success

If the operation is successful, this tool takes the Success exit path.

Failure

This tool takes the Failure exit path if the syntax of the Where clause is incorrect or if the CIC Data Source is configured incorrectly.

Notes

1. To prevent updating the primary key (which is not allowed by many RDBMSs, even if the value will not change) this tool uses the ODBC API *SQLPrimaryKeys*. Although this API is supported by most ODBC drivers, it is a Level 2 extension API and therefore you may run into an occasional driver that doesn't support it. If your ODBC driver doesn't support *SQLPrimaryKeys*, this tool will still work, but you must manually remove the primary key attribute(s) and value(s) from the input lists when updating. When inserting, the primary key data must still be present (unless your primary key is an auto-increment type – see note 2 below).
2. If your primary key is an auto-increment type you must exclude it from inserts, which means you can just exclude it always. Note that the SQL Server *uniqueidentifier* type is not included here, since it must be supplied as data (usually with the *NEWID()* function) during the insert.
3. Currently, the only non-updateable data type that is recognized and automatically excluded is the SQL Server *timestamp* type. If your RDBMS has other types that are not updateable, you will need to manually exclude them (and please report these to Genesys so that they can be incorporated into future releases).
4. *SQLPrimaryKeys* requires, as inputs, a catalog name and a schema name. Usually, but not always, the catalog will correspond to the database name; the schema name will usually, but not always, correspond to the owner/creator of the table (the "qualifier"). By default, this tool will use the *qualifier* entry of the CIC data source for the schema, and it will look for a DB= or a DATABASE= entry in the *Additional Information* entry for the catalog. However, *SQLPrimaryKeys* has some obscure semantics regarding NULL vs. empty ("") strings for the catalog and schema names. Basically, you would use NULL as a wildcard to mean all catalogs or all schemas. Unless the table name is unique in a catalog (or across all catalogs if NULL is used for the catalog), this is clearly a problem for this tool, since it is expected to update exactly one table. The use of empty strings is fairly rare, and an empty schema means those tables that do not have schemas, and an empty catalog means those tables that don't have catalogs. A problem arises, however, because currently an empty/missing *qualifier* CIC data source entry means to use the default (e.g. "dbo"). The same applies to a missing database entry in the *Additional Information* field. To solve this problem, this tool will first look for an optional CATALOG= and a SCHEMA= entry in the *Additional Information* field that will be used to **override any qualifier entry or database entry**. To denote a NULL catalog or schema input, use the keyword NULL (e.g. CATALOG=NULL;); to denote an empty catalog or schema input, omit any characters (e.g. SCHEMA=;). As another example, if the catalog does not correspond to the database (or you don't feel like adding a database entry to the *Additional Information*), you can specify it explicitly (e.g. CATALOG=mkgt2;).

DB Query

This Database tool retrieves a result set from a table based on a Where clause. The result set retrieved matches the conditions you specify in your Where clause. You can then specify a variable (bind) for each row of data returned.

The following example shows you one example of a where clause you might want to construct, and then shows you how to configure DB Query to execute that where clause.

Sample Where Clause

You have a database containing transaction IDs, the names of the Agent's who processed the transactions, the Agent's phone number, and the cost of the transaction. You want to retrieve all purchase records for a single agent. In this example, the agent's name is specified in a variable called strAgent1. The Clothing database to be queried is shown below.

ID	Name	Phone	Cost
123	Bill	555-1234	\$5.00
124	Bill	555-1234	\$50.00
125	Monica	555-9876	\$10.00
126	Bill	555-1234	\$15.00
127	Monica	555-9876	\$5.00
Empty	Empty	Empty	Empty

The Where clause you should use to retrieve the value is shown below:

```
SELECT ID, Cost FROM Clothing WHERE Name='strAgent1'
```

Creating the Where Clause with the DB Query Tool

Now that you know the Where clause you want to use, follow these steps to configure the DB Query tool to execute the Where clause.

1. Specify the **Connection Variable** and **Data Source**.

The Connection Variable is generated by the DB Connect step elsewhere in the handler. The Data Source list is populated with ODBC data sources you have configured. For more information on these two parameters, see the field documentation further down in this help topic.

2. Select a **Table Name**.

Once you have selected a data source, the Table Columns for that data source are listed as choices for this parameters. Using the our sample database as an example, Clothing is the table listed. The Table Name forms the **FROM** portion of your Where Clause, so you've now configured **FROM Clothing**.

Once you have selected a Table Name, the columns within that table appear in the Table Columns List.

3. In the **Table Columns list**, select the columns you want to query.

When you pick Clothing as your Table Name, ID, Name, Phone, and Cost all appear in the Table Column list. You would highlight ID, then click the Bind... button. In the Column Binding Dialog, you would select the variable to which will receive the value retrieved from the ID column. Next you would select the Cost table column and bind the variable that will receive the value of from the Cost Column. (The variable to which the Table Columns are bound are created with [Assignment](#) steps that precede this DB Query step.)

Binding variables to the ID and Cost columns completes the **SELECT** portion of your Where clause. Now you have configured DB Query to execute the following statement:

```
SELECT ID, Cost FROM Clothing
```

The next step will be to add the **WHERE** portion of the Where clause.

Note: Interaction Center 4.0 supports SQLServer 2008, which supports a new datetime2 data type. To bind this type to a datetime variable in this and other toolsteps, use the latest driver "SQL Server Native Client 10.0" in the DSN. Otherwise, Interaction Designer won't recognize the datetime2 data type and will treat it as a string.

4. Specify the **Where Clause Variable**.

In the Where Clause Variable parameter, you will specify the variable that contains your Where clause. You must create your Where clause in an [Assignment](#) step preceding this DB Query step.

The DB Query step will insert the **WHERE** portion for you, so your Assignment step only needs to assign:
Name = 'strAgent1'

To type this value in Expression Editor Assistant, you would type the following:

```
"Name=" & SQLStr(strAgent1)
```

When the handler executes, this expression results in **Name = 'strAgent1'**. Once you have assigned this value to a string variable, you can select that variable from the DB Query's Where Clause Variable drop-down list.

Once you have selected your Where Clause Variable, the **WHERE** portion of your Where clause is complete. When you run the handler the following Where clause is executed:

```
SELECT ID, Cost FROM Clothing WHERE Name='strAgent1'
```

5. After this step executes, you can use a [DB Fetch](#) step to retrieve, one record at a time, the records stored in the result set. This example is continued in the documentation for that tool.

Settings Page

Connection Variable

Specifies what connection on which the query is performed. Use a connection variable that you specified in a [DB Get Connection](#) step. Each connection can contain only one result set, so if you need more than one result set, you should open more than one connection.

Data Source

Select a data source to query. The selected data source's table columns appear in the Table Name drop-down list box.

Table Name

This is a list of tables available for the data source you specified in the previous Data Source parameter. When you select a Table Name, the columns defined in that table appear in the TableColumns/VariableList.

Runtime Table Name

Specify a string expression that, at handler runtime, is used as the table name for the select statement. This field is optional. If no value is specified, the table selected in the Table Name field will be used.

Timeout (seconds)

The number of seconds to wait for this operation to complete successfully before taking the failure path. Any value less than or equal to zero results in the default value of 60 seconds being used. You may specify a decimal value, such as 5.5.

Table Columns/Variable List

This is a list of Columns for the table specified in the previous Table Name parameter. You can bind variables created previously in [Assignment](#) steps earlier in the handler.

Bind Button

A binding specifies which variable receives the value of the selected column and row. Use this button to bind a column to a variable. When you execute this query and do a fetch, that variable receives the value of that column for the current row. A second fetch overwrites the value of this variable.

Note: The bindings dialog box allows bindings to "convertible" types. This is useful if you want to read a value from a column and have it be converted into another type automatically. For instance, SQL integers into numeric type variables.

Users wanting to bind a variable to a column are presented with the list of variables that are eligible for binding to the SQL type for that column. In past releases, these variables would have been all the same type (i.e. there was only one handler variable type allowed for a given SQL type). Now the list of variables will contain the variables of all the types that can be converted from the SQL Type.

Unbind Button

Use this button to unbind a column from a variable. See *Bind Button* above.

Where Clause Variable

Specify the variable that contains a Where clause. You will need to construct the Where clause in an Assignment step preceding this step. For more information on building Where clauses, see [Sample Where Clauses](#).

Exit Paths

Success

If this step executes successfully, this step takes the Success exit path.

Failure

If this step does not execute successfully, this step takes the Failure exit path. This can occur when information specified about the table or view is incorrect, the DB connection variable is not valid, or the Where clause was not correct (i.e. it did not contain valid SQL statement).

DB Release Connection

This Database tool releases a connection to the database server, freeing up the server for another connection. This step is usually followed by a [DB Close](#) step.

You may not want to follow this step with a DB Close in cases where you have opened multiple connections to the same data source. This is useful in cases where you want to perform "nested" queries against the database.

For instance, suppose you execute a query step, then fetch a record, then you need to perform an additional query (based on some data in that fetched record) in order to process the record. If you perform that 2nd query on the same connection, your first result set will be lost (the server keeps 1 current result set for each connection). So, you open another connection (or use another one that was opened before) and perform the 2nd query on that. Then you fetch the result from that 2nd query and finish whatever processing you needed to do for that original record. Now you can go back and fetch the 2nd record from the 1st query. To process the 2nd record you'll probably do the same thing you did for the first – do another query & fetch, then finish processing.

After you finished processing all the records from your original query, you're left with 2 open connections. If you only need one for the rest of your handler, then you can release the 2nd one.

Settings Page

Database Connection Variable

Specifies which database variable for the connection you want to release.

Exit Paths

Success

If this step executes successfully, this step takes the Success exit path.

Failure

If this step does not execute successfully, this step takes the Failure exit path. This can occur when the DB connection variable is not valid.

DB SQL Exec

This Database tool is useful when you want to do something with a database other than query. DB SQL Exec can be used to insert data into a database, update a row already in a database, delete data from a database, or run a stored procedures on the server (scripts stored on the database that operate on the data tables).

To use this tool, build a SQL statement using one or more steps created with the Assignment Tool, and pass it in as a string variable. Use this string variable as your SQL Statement Variable on the next page. You can use the & (string append) function in several Assignment steps to build complex SQL statements.

Settings Page

Database Connection

The name of the database connection you created in a DB Get Connection statement.

SQL Statement Variable

The name of the variable that contains the SQL statement. The value of this variable is created with one or more [Assignment](#) steps. For more information on building SQL statements, see [Sample SQLExec Statements](#).

Timeout (seconds)

The number of seconds to wait for this operation to complete successfully before taking the failure path. Any value less than or equal to zero results in the default value of 60 seconds being used. You may specify a decimal value, such as 5.5.

Exit Paths

Success

If this step executes successfully, this step takes the Success exit path.

Failure

If this step does not execute successfully, this step takes the Failure exit path. This can occur when the DB connection variable is not valid, or the SQL statement is not valid.

DB SQLExec2

This database tool allows for the execution of any SQL statement that does not return a result set. If used for queries or stored procedure calls that return result sets, you will not be able to retrieve the results. Use [DB Query](#) and [DB Stored Procedure](#) for stored procedure calls that return result sets instead of this tool.

This tool will optionally allow for the retrieval of the number of affected rows (INSERTs, UPDATEs, and DELETEs only).

Like the DB list tools, this tool will grab a connection from the cache (or create a new one if one is not available), perform the operation, then release the connection back to the cache. In other words, it is completely self-contained, which means you do not need to call [DB Open](#) & [DB Get Connection](#) prior to using this tool.

Inputs

IC Data Source

The name of the CIC (*not* ODBC) data source to use.

SQL Command

The SQL command to execute.

Timeout (seconds)

The time, in seconds, before this tool will return with a timeout error. Where supported by the RDBMS, the statement/query timeout on the server will also be set with this value.

Check Count of Affected Rows

If TRUE, a call will be made to retrieve the count of the number of rows affected by this command.

If you know the command does not affect any rows or produce a row count (or you don't care about the row count) set this to FALSE as an optimization.

Outputs

Affected Row Count

If the input "Check Count of Affected Rows" is TRUE, this will contain the row count; note that a row count of zero is possible in many situations (e.g. the where clause of an UPDATE does not specify any rows, the command does not produce a row count, etc.). If "Check Count of Affected Rows" is FALSE, or an error was encountered, this value will be set to -1.

Error Code

If the tool execution was successful, this will be set to zero; otherwise, it will be set to the native (RDBMS) error code, or in the case of an internal error (e.g. Notifier error) it will be set to an CIC-specific error code.

Exit Paths

Success

The operation was successful. Note that this doesn't mean that there were affected rows; it only means the SQL statement itself did not generate any errors.

No Rows

This path will only be taken if all three of the following conditions are met:

1. There were no errors
2. The input "Check Count of Affected Rows" is TRUE
3. The result did not affect any rows (or did not produce a row count).

Failure

There was a SQL or internal error generated while trying to execute this command. Check the IPDBServer.log for information on the error.

DB Stored Procedure

This Database tool calls a stored procedure on your database server. This tool's properties allow you to bind handler variables/expressions to the stored procedure parameters, result columns, and return value. After this tool executes, you can use [DB Fetch](#) to fetch result rows (if necessary). Depending on which database server product you are using, the output parameter values and the return value will either be populated:

- a. immediately on the call to DB Stored Procedure, or
- b. if a result set is returned, output parameters and return value won't be set until all rows have been fetched. This is for all databases.

Note: For stored procedures that return a result set, you must wait until after all of the rows in the result set are processed to check output parameters and return values. This is a standard ODBC requirement.

Consult with your database administrator to determine the behavior of your particular database server.

Related Topics

[DB Stored Procedure Definition dialog](#)

What is a stored procedure?

A stored procedure is a function or script that is saved on a database server. This script can be executed on the database server at the request of a database client. Stored procedures can receive input from the client at the time they're executed. Stored procedures carry out their processing on the server and can return data/results of that processing to the client. Database administrators use stored procedures to carry out complex functions that are best performed on the server (as opposed to the client). In addition, stored procedures often outperform normally submitted SQL because the SQL in the procedure is pre-parsed, and execution plans pre-generated.

A definition of a stored procedure specifies parameters, result set, and the return value. A stored procedure parameter is similar to the standard programming concept of a function parameter – these parameters can be input data, meaning the value passed in is used during processing, or output data, meaning the reference to a value will be "filled in" during processing, or they can be both, meaning their initial value will be used during processing, and that value may then be changed before processing completes. Result sets are the same concept as for [DB Query](#) – the stored procedure can execute a query during its processing, and the client will be able to access the result set of that query by calling fetch for each row in the result set. The stored procedure return value is similar to an output parameter – it's a value that will be "arrived at" during processing and will be returned to the client.

An example of how this tool is used:

First, a stored procedure must be created. For MS SQL Server, this can be done using the SQL Server Query Analyzer: Open the Analyzer, type the Stored Procedure's contents, and click on the Execute button (Green arrow). The stored procedure is created.

Here's an example of a stored procedure created for testing purposes:

```
CREATE PROC makeatable
@lastcalltext char(20) output,
@totalrows int output
as
declare @lastcall datetime
drop table test1234
SELECT calldetail.* INTO test1234
FROM calldetail
WHERE (((calldetail.siteid)="101"));
UPDATE test1234 SET test1234.siteid = "991";
SELECT @lastcalltext = max (InitiatedDate) from test1234
SELECT @totalrows=COUNT(*)
FROM test1234
print @lastcalltext
print @totalrows
return @totalrows
```

Notes:

1. The Stored Procedure name is makeatable.
2. Makeatable is designed to work on the same SQL Server database CIC uses to store its call data (the database referred to by the CIC Server EIC_Tables ODBC DSN).
3. The stored procedure, as written, assumes that the CIC Server's site identifier is 101 (to find out what the Site ID is for a given CIC Server, open Interaction Administrator, click on the Production container and double-click on Configuration item on the right. You can change the 101 in the example to whatever the site ID is for the CIC Server under test.

4. Makeatable takes 2 parameters: @lastcalltext, a string, and @totalrows, an integer. It needs to receive these 2 parameters on input, but doesn't use them. It outputs the date and time of the last call logged in the database (@lastcalltext), and the amount of calls that server processed (@totalrows).

makeatable will delete a table called test1234 (without warning), if it exists, so make sure you don't have any valuable data in such a table.

5. Before using a stored procedure in Interaction Designer, it's a good idea to make sure it runs properly on the SQL Server. So, open the SQL Server Query Analyzer, type

```
makeatable "blah", 5
```

and click on the execute button. The output should display the number of rows affected (twice) followed by the date of the last call in the database, followed by some integer (number of calls, same as the number of rows affected).

If this worked, the test Stored Procedure can be used in Interaction Designer. Here's how:

1. Create a new Handler in Interaction Designer. Use the SendCustomNotification initiator so that you can launch the handler from a command line.
2. Use two Assignment steps to initialize the 2 variables mentioned in step 4 of the previous list: A string variable, which will be bound to @lastcalltext, and an integer variable, which will be bound to @totalrows. Assign "blah" and 0 to the 2 variables, respectively (it really doesn't matter what's in there, as long as the data type is correct).

Note: You can also declare new variables directly from the DB Stored Procedure tool created in Step 5 of this procedure. After you click the Bind button, you can declare a new variable from the Bind dialog that appears.

3. Use a DBOpen step to open ODBC DNS eic_tables. Use a user ID of eic_user, and a password of i3 (unless the default password has been changed).
4. Use a DBGetConnection step.
5. In a DB Stored Procedure step, select the stored procedure. To do this, you'll pick the Data Source Name (EIC_Tables), Schema (that's the procedure creator/owner, usually dbo), and the Stored Procedure (makeatable). If this works properly, @lastcalltext and @totalrows should automatically appear in the parameters window. Bind variables to these parameters to receive the values output from the stored procedure.

Note: If your parameters don't appear in the parameters window, you may need to manually modify CIC's definition of the stored procedures in the [DB Stored Procedure Definition](#) dialog.

6. Add any additional desired logic to the handler, save it, publish it, and activate it.

Note: If you batch publish a handler with this tool and use the /LogBatchPublish command line argument, the publish process may display a publish dialog. See the [Batch Publish](#) topic for more information.

7. Start the handler from the command line using the sendcustomnotification command.

When the handler runs, your stored procedure is run on the DB server.

Parameters Page

DB Connection

The variable containing the database connection generated by the [DB Get Connection](#) tool.

DSN (Catalog =)

The name of the data source on which the procedure was created. The concept of the catalog varies from one RDBMS vendor to the next. It usually corresponds to the physical database, but there are exceptions. For example, with dBase & FoxPro it usually corresponds to the directory in which the data files reside. While some RDBMSs allow you to dynamically which catalog contexts within a connection, CIC currently doesn't support this, and assumes the catalog is tied to the ODBC DSN. So, if access to multiple catalogs is required, you must create and configure multiple ODBC DSNs.

Schema

The schema in which the stored procedure resides. The "schema" is the official ANSI term for what used to be commonly known as

the "qualifier". It is usually the creator/owner of the database object. Worth noting is that Sybase and MS SQL Server have a special schema name that is used when the owner of the object is also the *database owner*: "dbo".

Stored Procedure

The name of the stored procedure as it is stored on the DB server.

Note: You cannot change actual name as it's defined on the DB server from this field.

Runtime Stored Procedure

An optional expression that results in a string that names a stored procedure. This allows you to run a stored procedure dynamically from a handler. If you specify a value here, the any value in the Stored Procedure parameter is ignored.

Note: If specifying a value here, you are responsible for including any required schema prefix. For example, if the stored procedure is called 'MyProc', and it is owned by user 'Fred' who is *not* the database owner (dbo), **and** the connection was created under a user other than Fred, then you would need to provide 'Fred.MyProc' as the value here. As another example, if Fred *was* the dbo, then you could supply either 'dbo.MyProc' or simply 'MyProc', since Sybase and MS SQL server will automatically attempt to resolve any unqualified object name with 'dbo'. As a final example, if the database connection was created using Fred's account, then either 'Fred.MyProc' or 'MyProc' will work.

Execution String

Optional. Any executable SQL statement can be provided. If the statement contains embedded parameter markers ('?'), they will be bound to the corresponding bind parameters that are specified (it is up to the user to make sure the number of parameter markers matches the number of bound parameters, and that the ordinal positions are correct). The intent is to allow a way to provide the RDBMS-specific stored procedure call syntax in the event that the ODBC driver doesn't support the ODBC escape syntax for stored procedure execution. Fortunately, since most drivers nowadays support the escape syntax, you should not normally need to provide anything here.

Note: Although not originally intended as a feature, the execution string can be any valid SQL – it is not limited to a stored procedure call. For example, you could use it to perform a SELECT Count(*). You can pass input parameters and also get a result set; however, you cannot set output parameters or get a procedure return (since there is no procedure).

Timeout (seconds)

The number of seconds to wait for this operation to complete successfully before taking the failure path. Any value less than or equal to zero results in the default value of 60 seconds being used. You may specify a decimal value, such as 5.5.

Parameters list

The parameters for the specified stored procedure. You can bind variables to the input and input/output parameters. If you do not see all of the parameters that should be listed, your ODBC driver may not be passing all of the parameters to this tool. In this case you may need to define these parameters in the [DB Stored Procedure Definition](#) dialog (which you can open through the Utilities menu).

Note: In some cases you may have to take a couple of extra steps to convert values in a handler to parameters in the stored procedure. For example, you have a SMALLINT data type as an input parameter that is basically acting as a Boolean (i.e. it will either hold a 1 or a 0). When you open the expression editor to supply a value, the expression editor only displays the primary mapping type for SMALLINT, which is an CIC integer. This is because the expression editor is designed with the notion that everything is with respect to the current (single) data type. Unlike Interaction Designer, it has no way of presenting a list of both Boolean and integer variables. To work around this problem, define an integer variable to use, then set it with the Boolean value using one of the expression editor's type conversions.

Bind button

Click the Bind button to bind the selected parameter to a variable (for in/out and out parameters) or expression (for input parameters).

Unbind button

Click the Unbind button to unbind a parameter from a variable or expression.

Clear Bindings button

Click the Clear Bindings button to remove all bindings to all parameters.

Stored Procedure Return Value

A value returned as a result of the processing performed by the stored procedure. The type of value returned depends on the RDBMS. For example, Oracle and DB2 allow you to return most legal SQL data types (LONG or BLOB types usually being the exceptions), but Sybase and MS SQL Server both limit return types to INTEGERS. RDBMSs don't typically allow complex types, like lists, to be returned. The handler must include a condition to inspect the return value.

Result Set Page

This list is populated if the stored procedure fetches any data. You can bind variables to the fetched data. If you do not see all of the column names that should be listed, your ODBC driver may not be passing all of the column names to this tool. In this case you may need to define these column names and other result set properties in the [DB Stored Procedure Definition](#) dialog (which you can open through the Utilities menu).

Note: CIC does not currently support retrieving multiple result sets. If the stored procedure generates multiple result sets, then only the first result set can be retrieved.

Note: For stored procedures that return a result set, you must wait until after all of the rows in the result set are processed to check output parameters and return values. This is a standard ODBC requirement.

Name

The name of the column from which the data is fetched.

Position

The ordinal position of the column in the result set, starting from 1.

Expression

This is the variable to which the fetched data is bound. You can specify only a variable, not an expression.

Bind... button

Click the Bind button to bind the selected column to a variable

Unbind button

Click the Unbind button to unbind a variable from a column.

Clear Bindings... button

Click the Clear Bindings button to remove all bindings to all columns.

Exit Paths

Success

This step takes the Success exit path if the ODBC call succeeds, regardless of the value that is returned by the stored procedure.

Failure

This step takes the Failure exit path if the ODBC call fails. Probable causes include:

- Stored procedure not found. Usually this is due to a wrong or missing schema qualifier, or insufficient privileges.
- Insufficient execution privileges.
- Data type mismatch with one of the procedure parameters or return value.
- To troubleshoot the cause of a failure, the most important thing is to examine the IPDBServer.log file – it will normally (well, hopefully) contain a useful RDBMS error message.

Dialer

Dialer Call Completed

This Dialer tool disposes a Dialer call.

Inputs

Campaign ID

The unique ID of the call's Dialer campaign.

Call ID

The unique for the call.

Wrapup Code

A string that indicates the completion status of a call step. Wrap-up codes appear in scripts and allow agents to disposition calls by choosing a call outcome for a campaign. For reporting purposes, each user-defined wrap-up code must map to a wrap-up category.

Wrap-up Category	Default wrap-up Codes (user-defined)
Default inbound category	None
Ambiguous	Ambiguous Ambiguous - Agent Connection Broken Ambiguous - Agent Logout Ambiguous - Agent Received a New Call
Busy	Busy Busy - Busy Signal Busy - Disconnect before Analysis Busy - Not Reached Busy - Remote Busy
Deleted	Deleted Deleted - Do Not Call
Failure	Failure Failure - Timeout

Fax	Fax Fax - System Hang up on Fax
Machine	Machine - Answering Machine Machine - Failed to play recording Machine - Machine Machine - Recording played to Fax Machine - Recording played to Machine
No Answer	No Answer No Answer - Answering Machine No Answer - Disconnect before Analysis No Answer - No User Responding No Answer - Timeout No Answer - User Alerting No Answer
No Lines	No Lines No Lines - No IP Response
Non-Dialer Call	Non-Dialer call
Not Reached	Not Reached Not Reached - Disconnect before Analysis
Phone number deleted	Phone number deleted
Phone number success	Phone number success
Policy Scheduled	Policy Scheduled
Remote Hang Up	Remote Hang Up Remote Hang Up - Contact Hang Up Remote Hang Up - Remote Hang Up after Transfer Remote Hang Up in Attendant
Rescheduled	Rescheduled
Scheduled	Scheduled Scheduled - Callback
SIT	SIT
SIT Callable	SIT Callable - Disconnect before Analysis SIT Callable - Ineffective Other (see note that follows table) SIT Callable - No Circuit SIT Callable - No Route to Destination SIT Callable - Normal SIT Callable - Protocol Error SIT Callable - Reorder SIT Callable - Temporary Failure

SIT Uncallable	SIT Uncallable - Bad Number SIT Uncallable - Disconnect Before Analysis SIT Uncallable - Ineffective Other SIT Uncallable - Invalid Number Format SIT Uncallable - No Circuit SIT Uncallable - No IP Response SIT Uncallable - Number Changed SIT Uncallable - Reorder SIT Uncallable - Unassigned Number SIT Uncallable - Unknown Tone SIT Uncallable - Vacant Code
Skipped	Skipped - Agent Skip Skipped - Do Not Dial Skipped - Policy No valid Phone Number
Success	System Hang Up on Fax System Hang Up on Live Voice System Hang Up on Machine Success Success - Recording played to Fax Success - Recording played to Live Voice Success - Recording played to Machine
System Hang Up	System Hang Up System Hang Up - Agent not available for callback System Hang Up - Attendant Transfer failed System Hang Up - Failed to play recording System Hang Up - Failed to route call to agent System Hang Up - Failed to send fax System Hang Up after Transfer System Hang Up in Attendant
Transferred	Transferred
Wrong Party	Wrong Party Wrong Party - Wrong Number

Exit Paths

Next

This tool step always takes the Next exit path.

Get Campaigns

This Dialer tool returns a list of campaign IDs and names that correspond to the active, currently running campaigns.

Caution: Do not use this tool in your handlers as it may cause CIC to function incorrectly. It is intended for Interaction Dialer only. You will never need to use or modify the values set in this tool.

Inputs

Get even campaigns that are not running?

Outputs

Campaign Name List

List of campaign names that correspond to currently running campaigns.

Campaign ID List

List of campaign IDs that correspond to currently running campaigns.

Exit Paths

Success

This tool step takes the Success exit path if...

Failure

This tool step takes the Failure exit path if....

Get Dialer Attributes

This Dialer tool gets the attributes for a Dialer call.

Note: This tool only appears in the tool palette if Dialer is installed.

Inputs

Campaign ID

The ID of the campaign for the call.

Call ID

The call ID for the call.

Outputs

Attribute Name List

The names of the Dialer attributes on the call.

Attribute Value List

The values of the Dialer attributes on the call.

Exit Paths

Success

This tool takes the Success exit path if the attribute values were retrieved successfully.

Failure

This tool takes the Failure exit path if the attribute values were not retrieved because the call was not found.

Get Logged In Agents

This Dialer tool retrieves the currently logged in agents associated with a specified campaign.

Note: This tool only appears in the tool palette if Dialer is installed.

Inputs

Campaign ID

The ID of the campaign being queried.

Outputs

Agent ID List

List of strings containing the IDs of all currently logged in agents associated with this campaign.

Exit Paths

Next

This tool always takes the Next exit path.

Place Dialer Call

This Dialer tool tells Interaction Dialer whether or not to place a given call.

Notes: This tool only appears in the tool palette if Dialer is installed.

Important: Use caution when setting the `bDialerPlaceCall` variable to false in order to stop Dialer from placing a call. Dialer expects a disposition to clear out a contact that it called or expected to call. If you set `bDialerPlaceCall` variable to false, use the [Dialer Call Completed](#) tool to disposition calls so that Dialer properly deallocates the call.

Inputs

Campaign ID

The identifier for the call's campaign.

Call ID

The unique identifier for this call.

Wrapup Code

A string that indicates the completion status of a call step. Wrap-up codes appear in scripts and allow agents to disposition calls by choosing a call outcome for a campaign. For reporting purposes, each user-defined wrap-up code must map to a wrap-up category.

Wrap-up Category	Default wrap-up Codes (user-defined)
Default inbound category	None

Ambiguous	Ambiguous Ambiguous - Agent Connection Broken Ambiguous - Agent Logout Ambiguous - Agent Received a New Call
Busy	Busy Busy - Busy Signal Busy - Disconnect before Analysis Busy - Not Reached Busy - Remote Busy
Deleted	Deleted Deleted - Do Not Call
Failure	Failure Failure - Timeout
Fax	Fax Fax - System Hang up on Fax
Machine	Machine - Answering Machine Machine - Failed to play recording Machine - Machine Machine - Recording played to Fax Machine - Recording played to Machine
No Answer	No Answer No Answer - Answering Machine No Answer - Disconnect before Analysis No Answer - No User Responding No Answer - Timeout No Answer - User Alerting No Answer
No Lines	No Lines No Lines - No IP Response
Non-Dialer Call	Non-Dialer call
Not Reached	Not Reached Not Reached - Disconnect before Analysis
Phone number deleted	Phone number deleted
Phone number success	Phone number success
Policy Scheduled	Policy Scheduled
Remote Hang Up	Remote Hang Up Remote Hang Up - Contact Hang Up Remote Hang Up - Remote Hang Up after Transfer Remote Hang Up in Attendant

Rescheduled	Rescheduled
Scheduled	Scheduled Scheduled - Callback
SIT	SIT
SIT Callable	SIT Callable - Disconnect before Analysis SIT Callable - Ineffective Other (see note that follows table) SIT Callable - No Circuit SIT Callable - No Route to Destination SIT Callable - Normal SIT Callable - Protocol Error SIT Callable - Reorder SIT Callable - Temporary Failure
SIT Uncallable	SIT Uncallable - Bad Number SIT Uncallable - Disconnect Before Analysis SIT Uncallable - Ineffective Other SIT Uncallable - Invalid Number Format SIT Uncallable - No Circuit SIT Uncallable - No IP Response SIT Uncallable - Number Changed SIT Uncallable - Reorder SIT Uncallable - Unassigned Number SIT Uncallable - Unknown Tone SIT Uncallable - Vacant Code
Skipped	Skipped - Agent Skip Skipped - Do Not Dial Skipped - Policy No valid Phone Number
Success	System Hang Up on Fax System Hang Up on Live Voice System Hang Up on Machine Success Success - Recording played to Fax Success - Recording played to Live Voice Success - Recording played to Machine

System Hang Up	System Hang Up System Hang Up - Agent not available for callback System Hang Up - Attendant Transfer failed System Hang Up - Failed to play recording System Hang Up - Failed to route call to agent System Hang Up - Failed to send fax System Hang Up after Transfer System Hang Up in Attendant
Transferred	Transferred
Wrong Party	Wrong Party Wrong Party - Wrong Number

Place Dialer Call?

Boolean indicating whether or not the call should be placed.

Exit Paths

Success

This tool takes the Success exit path unless one of the failure criteria is met.

Failure

This tool takes the Failure exit path if the call cannot be placed or its completion cannot be processed due to one or more of the Place Dialer Call Inputs.

Restart Campaign

This Dialer tool resets the call list in a specified campaign.

Note: This tool only appears in the tool palette if Dialer is installed.

Inputs

Campaign ID

The identifier for the campaign to reset.

Exit Paths

Success

This tool takes the Success exit path if the campaign is successfully reset.

Failure

This tool takes the Failure exit path if the campaign cannot be reset.

Set Dialer Attributes

This Dialer tool sets the Dialer attributes for a Dialer call.

Note: This tool only appears in the tool palette if Interaction Dialer is installed.

Inputs

Campaign ID

The identifier of the call's campaign.

Call ID

The call ID for the call.

Attribute Name List

The names of the Dialer attributes to be set.

Attribute Value List

The values to set for the Dialer attributes.

Exit Paths

Success

This tool takes the Success exit path if the attribute values were set successfully.

Failure

This tool takes the Failure exit path if the attribute values were not set because the call was not found.

Director

Director Tools and Initiators in Interaction Designer

In Interaction Designer, tools for developing handlers with Interaction Director functionality appear on a design palette tab named "Director." The supported tools include:

- [Director Monitored Server Ready tool](#)
- [Director Select Queue Tool](#)
- [Play Audio File \(No Conference\) tool](#)
- [Send ATT Announcement Code Tool](#)
- [Send ATT Post Feature Code Tool](#)
- [Send MCI DDD Tool](#)
- [Send MCI Destination w/DNIS Override Tool](#)
- [Send MCI Error Tool](#)
- [Send MCI IDDD Tool](#)
- [Send MCI Use Default Tool](#)
- [Send Route Command Tool](#)
- [Send SIP Destination](#)
- [Send Simulated Route Command Tool](#)
- [Send Sprint Error Tool](#)
- [Send Sprint Reject Tool](#)
- [Send Sprint Use Default Tool](#)

Some tools appear on the Director tool palette but are reserved for internal use only or by special instructions from PureConnect Customer Care. These include:

- [Director Get Overflow Spec](#)
- [Director Get Time String tool](#)
- [Director Interaction Removed tool](#)
- [Director Interaction Transferred tool](#)
- [Director Process Offering Interaction tool](#)
- [Director Remote Audio Path Complete tool](#)
- [Director Remote Routing tool](#)
- [Director Route Email tool](#)
- [Director Send Message Tool](#)
- [Director Send Response Tool](#)

Interaction Director initiators are accessible via the File menu. These include:

- [Accept Generic Route Request initiator](#)
- [Accept MCI Route Request initiator](#)
- [Accept Sprint Route Request initiator](#)
- [Accept ATT Route Request initiator](#)
- [Accept Generic Route Request initiator](#)
- [Accept Generic Routing Statistics initiator](#)
- [Director Connection Lost initiator](#)
- [Director Connection Restored initiator](#)

Director Monitored Server Ready tool

This Director tool is only for use in conjunction with Director and must appear at the end of the DirectorMs_ConnectionRestored handler.

Inputs

Response Correlation Id

An ID which identifies the message. This must be copied from the similarly-named output on the DirectorConnectionRestored initiator.

Response Destination Id

An ID which identifies the source of the message. This must be copied from the similarly-named output on the DirectorConnectionRestored initiator.

Outputs

None.

Exit Paths

Next

This path is taken when the operation completes.

Related Topics

 [Director Tools in Interaction Designer](#)

Director Select Queue Tool

This Director tool is intended to select a destination queue using the pre-call scoring criteria defined for the supplied list of Enterprise Groups.

Note: calling this tool affects the state of the system if a queue is returned.

Inputs

Interaction Type

The currently supported types are 0 (calls) and 5 (emails)

Universal ID

This should be a ID that would uniquely identify an interaction throughout its existence across all Director-connected servers

Sequence Number

This should be different every time the tool is called with a given Universal ID, but can be reset when the Universal ID changes.

Originating Server Name

This is an optional value that identifies the server that originated this request.

Originating Server Id

Optional value that is the site ID of the server that originated this request

Originating Interaction Id

This is an optional value that identifies any local interaction ID that the originating server might have assigned.

Enterprise Group Names

This is a list of Enterprise Group names that Director will examine in the order provided to find a suitable destination for the interaction. However, the only condition under which later Enterprise groups will be checked is if earlier ones result in no possible route. In other words, if a score can be calculated it will be. Only if it can't will Director move on to later Enterprise Groups. Another way of stating this is that the resulting destination queue will be the highest scoring one within an Enterprise Group (the first one that yields a score), not across all supplied groups.

Required Skills

This is a list of skill names that agents must possess to be considered candidates to receive the interaction

Minimum Skill Proficiency

This list, whose values must be parallel to the skill names list, specifies the minimum allowable proficiency level that agents must possess to be considered candidates to receive the interaction.

Maximum Skill Proficiency

This list, whose values must be parallel to the skill names list, specifies the maximum allowable proficiency level that agents can possess to be considered candidates to receive the interaction.

Send Data to Destination

This flag, if set to true, causes Director to send a message to the selected destination containing information regarding the assigned queue as well as the list of attribute names and values supplied in the inputs below. The information can be retrieved (via customization) when the interaction arrives at the destination.

Attribute Names

This list specifies the attribute names that should be sent to the destination server.

Attribute Values

This list, whose values must be parallel to the attribute names list, specifies the attribute values that should be sent to the destination server.

Outputs

Assigned Enterprise Group Name

If the success exit path is taken, this value will be the name of the Enterprise Group of which the selected queue was a member.

Assigned Queue Local Name

If the success exit path is taken, this value will be the name of the selected queue as defined on the Director configuration.

Assigned Queue Remote Name

If the success exit path is taken, this value will be the name of the selected queue as defined on the destination CIC server.

Assigned Server Name

If the success exit path is taken, this value will be the name of the destination CIC server

Assigned Server Id

If the success exit path is taken, this value will be the site ID of the destination CIC server

Assigned server Ip Address

If the success exit path is taken, this value will be the IP address of the destination CIC server server.

Exit Paths

Success

This path will be taken if a queue has been selected.

Failure

This path will be taken if an internal problem is encountered that prevented a valid evaluation of the Enterprise Groups supplied.

No Queue

This path means that the evaluation was completed however no destination queue could be selected due to routing criteria not being met in any supplied Enterprise Group

Related Topics

[Director Tools in Interaction Designer](#)

Play Audio File (No Conference) tool

This Director tool allows playing an audio file to both parties on a connected call without establishing a conference.

Inputs

Call Identifier

The call ID of the call to receive the Audio File play.

Audio File Name (.wav)

The name of a wav file to play.

Exit Paths

Success

This exit path is taken when the play operation completes.

Related Topics

[Director Tools in Interaction Designer](#)

Send ATT Announcement Code Tool

This Director tool sends an *announcement code* response to the ATT interface.

Inputs

Announcement Code

The code to be sent as specified by ATT for each customer

Customer Database Provided Digits

This is one or more numeric characters to be sent to the destination as allowed by the ATT specification.

Request Sequence Number

This value must be copied from the initiator that fires when a pre-call route request is received.

Respond To

This value must be copied from the initiator that fires when a pre-call route request is received.

Outputs

None.

Exit Paths

Success

This path is taken if the function was performed successfully.

Failure

This path is taken if an error occurred.

Related Topics

[Director Tools in Interaction Designer](#)

Send ATT Post Feature Code Tool

This Director tool sends a *post feature code* response to the ATT interface

Inputs

Post Feature Code

This is the code to be sent as specified by ATT for each customer.

Customer Database Provided Digits

This is one or more numeric characters to be sent to the destination as allowed by the ATT specification.

Request Sequence Number

This value must be copied from the initiator that fires when a pre-call route request is received.

Respond To

This value must be copied from the initiator that fires when a pre-call route request is received.

Outputs

None.

Exit Paths

Success

This path is taken if the function was performed successfully.

Failure

This path is taken if an error occurred.

Related Topics

[Director Tools in Interaction Designer](#)

Send MCI DDD Tool

This Director tool sends a *domestic digits to dial* response to the MCI interface.

Inputs

Domestic Digits to Dial

A telephone number to be dialed whose format is in accordance with the MCI specification.

Request Sequence Number

This value must be copied from the initiator that fires when a pre-call route request is received.

Respond To

This value must be copied from the initiator that fires when a pre-call route request is received.

Outputs

None.

Exit Paths

Success

This path is taken if the function was performed successfully.

Failure

This path is taken if an error occurred.

Related Topics

[Director Tools in Interaction Designer](#)

Send MCI Destination w/DNIS Override Tool

This Director tool sends a *destination with DNIS override* response to the MCI interface.

Inputs

Destination Label

This is a destination label that instructs the requesters routing logic what to do next. The format is in accordance with the MCI specification

DNIS Override

This is a DNIS value that will be presented to the destination.

Request Sequence Number

This value must be copied from the initiator that fires when a pre-call route request is received.

Respond To

This value must be copied from the initiator that fires when a pre-call route request is received.

Outputs

None.

Exit Paths

Success

This path is taken if the function was performed successfully.

Failure

This path is taken if an error occurred.

Related Topics

[Director Tools in Interaction Designer](#)

Send MCI Error Tool

This Director tool sends an *error* response to the MCI interface.

Inputs

Error Reason

This is an error code in accordance with the MCI specification.

Error Parameter

This is an error code in accordance with the MCI specification.

Request Sequence Number

This value must be copied from the initiator that fires when a pre-call route request is received.

Respond To

This value must be copied from the initiator that fires when a pre-call route request is received.

Outputs

None.

Exit Paths

Success

This path is taken if the function was performed successfully.

Failure

This path is taken if an error occurred.

Related Topics

[Director Tools in Interaction Designer](#)

Send MCI IDDD Tool

This Director tool sends an *international digits to dial* response to the MCI interface.

Inputs

International Digits to Dial

A telephone number to be dialed whose format is in accordance with the MCI specification.

Request Sequence Number

This value must be copied from the initiator that fires when a pre-call route request is received

Respond To

This value must be copied from the initiator that fires when a pre-call route request is received.

Outputs

None.

Exit Paths

Success

This path is taken if the function was performed successfully.

Failure

This path is taken if an error occurred.

Related Topics

[Director Tools in Interaction Designer](#)

Send MCI Use Default Tool

This Director tool sends a *use default* response to the MCI interface.

Inputs

Request Sequence Number

This value must be copied from the initiator that fires when a pre-call route request is received.

Respond To

This value must be copied from the initiator that fires when a pre-call route request is received.

Outputs

None.

Exit Paths

Success

This path is taken if the function was performed successfully.

Failure

This path is taken if an error occurred.

Related Topics

[Director Tools in Interaction Designer](#)

Send Route Command Tool

This Director tool sends a *destination* response to the interface from which the request was received.

Inputs

Destination Code

This value is the code to be returned in the response. This code tells the requester what to do next in its routing logic. Because this tool is used for to respond to of the pre-call carrier interfaces, the format and size of the string is determined by the individual carrier's specification.

Customer Database Provided Digits

This is one or more numeric characters to be sent to the destination as allowed by the carrier's specification.

Request Sequence Number

This value must be copied from the initiator that fires when a pre-call route request is received.

Respond To

This value must be copied from the initiator that fires when a pre-call route request is received.

Outputs

None.

Exit Paths

Success

This path is taken if the function was performed successfully.

Failure

This path is taken if an error occurred.

Related Topics

[Director Tools in Interaction Designer](#)

Send SIP Destination

This Director tool sends the necessary data back to the request originator so that it may route the incoming call to the desired destination.

This tool assumes that the destination is able to accept information in the SIP INVITE message. Only use this tool in configurations where the SIP INVITE has added information.

Inputs

Destination Domain

This string value should be set to the domain portion of the destination address. For example, "genesys.com".

Destination User Portion

This string value should be set to the user portion of the destination address. For example, "agenta".

Destination Queue Name

This string value should be set to the desired destination queue. This value would be obtained from Director Select Queue tool. This value is what exists on the destination machine, usually a CIC server, not the name of the queue item that points to that queue as configured on the Director machine in Interaction Administrator.

Attribute Names

A list of string values that contains names sent to the destination. This value is only used by a CIC server.

Attribute Values

A parallel list of string values that should be sent to the destination. This list must contain exactly the same number of items as the list of Attribute Names, or both lists will be ignored. This value is only used by a CIC server.

Respond To

This string value must be passed in unaltered from the Request Source parameter of the Accept SIP Route Request initiator. Director uses this value to determine which code module to send the response message.

Request Sequence Number

This integer value must be passed in unaltered from the Request Sequence Number parameter of the Accept SIP Route Request initiator. Director uses this value to associate the response with the original request.

Exit Paths

Success

This path is taken if the function was performed successfully.

Failure

This path is taken if an error occurred.

Related Topics

[Director Tools in Interaction Designer](#)

Send Simulated Route Command Tool

This Director tool sends a response back to the Simulate Call dialog. It is used only for testing the ability to run through a handler under development, to conveniently check the logic before allowing it to run in a live environment.

Inputs

Site

This should contain the CIC server that contains the assigned queue.

Queue

This should contain the assigned queue.

Estimated Wait

This can be filled in with an estimated wait time, if such a value is being calculated.

Additional Info

This field can optionally be used to show the routing information that would be sent back to the carrier. There is no specific format. Whatever is in this item is displayed on the dialog.

Request Sequence Number

This value must be copied from the initiator that fires when a pre-call route request is received.

Respond To

This value must be copied from the initiator that fires when a pre-call route request is received.

Outputs

None.

Exit Paths

Success

This path is taken if the function was performed successfully.

Failure

This path is taken if an error occurred.

Related Topics

[Director Tools in Interaction Designer](#)

Send Sprint Error Tool

This Director tool sends an *error* response to the Sprint interface.

Inputs

Error Reason

This is an error value whose format is in accordance with the Sprint specification.

Request Sequence Number

This value must be copied from the initiator that fires when a pre-call route request is received.

Respond To

This value must be copied from the initiator that fires when a pre-call route request is received.

Outputs

None.

Exit Paths

Success

This path is taken if the function was performed successfully.

Failure

This path is taken if an error occurred.

Related Topics

[Director Tools in Interaction Designer](#)

Send Sprint Reject Tool

This Director tool sends a *reject* response to the Sprint interface.

Inputs

Treatment Code

This is a response value whose format is in accordance with the Sprint specification.

Request Sequence Number

This value must be copied from the initiator that fires when a pre-call route request is received.

Respond To

This value must be copied from the initiator that fires when a pre-call route request is received.

Outputs

None.

Exit Paths

Success

This path is taken if the function was performed successfully.

Failure

This path is taken if an error occurred.

Related Topics

[Director Tools in Interaction Designer](#)

Send Sprint Use Default Tool

This Director tool sends a response back to the Sprint interface that indicates to use the Sprint default route for this request.

Inputs

Request Sequence Number

This value must be copied from the initiator that fires when a pre-call route request is received.

Respond To

This value must be copied from the initiator that fires when a pre-call route request is received.

Outputs

None.

Exit Paths

Success

This path is taken if the function was performed successfully.

Failure

This path is taken if an error occurred.

Related Topics

[Director Tools in Interaction Designer](#)

Director Initiators

Director Initiators - Monitored Server Only

- [Director Connection Lost initiator](#)
- [Director Connection Restored initiator](#)
- [Director Message Received initiator](#)

Director Initiators - Director Server Only

[Accept Generic Route Request initiator](#)

[Accept ATT Route Request initiator](#)

[Accept MCI Route Request initiator](#)

[Accept SIP Route Request initiator](#)

[Accept Sprint Route Request initiator](#)

[Accept Generic Routing Statistics initiator](#)

Accept Generic Route Request initiator

This initiator fires when a route request is received from any carrier for which an active interface has been configured on the Director server.

Initiator Properties

Notification Object Type

IDirector Routing

Object ID

{all} – this is the only choice. The values in the object id specify the carrier that sent the request.

Notification Event

Send Route Request – choose this for live operation. This is the event that is fired by the Director software to indicate that a route request has been received from the specified carrier.

Simulated Route Request – choose this for testing. This is the event that is fired by the Director software when a simulated route request is sent from the simulator dialog. In this way the actual handler that will be used in a live system can be tested for a given set of inputs.

Outputs

Dialed Number

This is the number that the carrier's request indicates was dialed by the caller.

Calling Number

This is the number that the carrier's request indicates the caller is calling from.

Customer Entered Digits

This is a set of digits collected in the carrier's network, if such a facility is available.

Enterprise Group List

This is the list of Enterprise Groups that have been configured in Interaction Administrator to be queried for the dialed number supplied in the output from Dialed Number, Calling Number, and/or Customer Entered Digits.

Key String

This is a sequence number that is guaranteed to be unique across all requests on a given Director server regardless of source or time received.

Request Source

This is an ID which identifies the source of the message. You must copy this to the similarly-named input on the any of the tools used to send a response back to the carrier.

Request Sequence Number

This is an ID which identifies the message. You must copy this to the similarly-named input on the any of the tools used to send a response back to the carrier.

Simulated Request

This is a Boolean value that indicates whether this request was sent from the simulator dialog.

Related Topics

 [Director Tools in Interaction Designer](#)

Accept MCI Route Request initiator

This initiator fires when a route request is received from MCI.

Initiator Properties Page

Notification Object Type

IDirector Routing

Object ID

{all} – do not choose this value or else requests from all carriers will come to the handler.

MCI – choose this value to receive only requests from MCI.

Notification Event

Send Route Request – choose this for live operation. It is the event that is fired by the Director software to indicate that a route request has been received from the specified carrier.

Simulated Route Request – choose this for testing. It is the event that is fired by the Director software when a simulated route request is sent from the simulator dialog. In this way the actual handler that will be used in a live system can be tested for a given set of inputs.

Outputs

Dialed Number

This is the number that the carrier's request indicates was dialed by the caller.

Calling Number

This is the number that the carrier's request indicates the caller is calling from.

Customer Entered Digits

This is a set of digits collected in the carrier's network if such a facility is available.

Enterprise Group List

This is the list of Enterprise Groups that have been configured in IA to be queried for the dialed number supplied in the output above.

Key String

This is a sequence number that is guaranteed to be unique across all requests on a given Director server regardless of source or time received.

Request Source

This is an ID which identifies the source of the message. You must copy this to the similarly-named input on the any of the tools used to send a response back to the carrier.

Request Sequence Number

This is an ID which identifies the message. You must copy this to the similarly-named input on the any of the tools used to send a response back to the carrier.

Simulated Request

This is a Boolean value that indicates whether this request was sent from the simulator dialog.

Related Topics

 [Director Tools in Interaction Designer](#)

Accept SIP Route Request Initiator

This initiator fires when an Interaction Director server receives a SIP-based pre-call route request. The handler that contains the initiator should make routing decisions. Once the handler has made a routing decision, the handler should send a response by using either the Send Route Command tool or the Send SIP Destination tool.

If the Send Route Command tool is used, the handler must format the destination address correctly. This destination address must be in the correct format, understood by the originator of the request, in order to redirect the incoming call request.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

External Handler Call

Object ID

Use "SIP".

Event ID

Use "Send Route Request"

Notification Event

SendRouteRequest

Outputs

Dialed Number

The string of the full called address of the incoming request. This value will conform to sip:user@DOMAINorIP:Port.

Calling Number

The string of the full calling address of the incoming request. This value will conform to sip:user@DOMAINorIP:Port.

Customer Entered Digits

This string value will be any digits collected or specified by the originator of the request. It may contain one or more possibly delimited items whose values and meanings are determined by the originator of the request.

Enterprise Group List

A list of strings specifying the configured Interaction Director Enterprise Groups that are associated with the Dialed Number in Interaction Administrator.

Key String

A guaranteed unique string that may be used as an key value in a database for data associated with a particular request.

Request Source

A string the Send SIP Destination tool uses. Do not change this value. The value describes which code module to send the response message.

Request Sequence Number

A integer that used by the Director tools. Do not change this value. The value associates the response with the original request.

Simulated Request

A boolean variable, that is "true" if the request did not originate from an external originator. A facility to send simulated requests is not currently implemented, so the value will always be "false".

Exit Paths

Start

This initiator always takes the Start exit path.

Accept Sprint Route Request initiator

This initiator fires when a route request is received from Sprint.

Initiator Properties

Notification Object Type

IDirector Routing

Object ID

{all} – do not choose this value or else requests from all carriers will come to the handler.

Sprint

Choose this value to receive only requests from Sprint.

Notification Event

Send Route Request – choose this for live operation. It is the event that is fired by the Director software to indicate that a route request has been received from the specified carrier.

Simulated Route Request – choose this for testing. It is the event that is fired by the Director software when a simulated route request is sent from the simulator dialog. In this way the actual handler that will be used in a live system can be tested for a given set of inputs.

Outputs

Dialed Number

This is the number that the carrier's request indicates was dialed by the caller.

Calling Number

This is the number that the carrier's request indicates the caller is calling from.

Enhanced Inquiry

This is a flag which indicates whether the request contains extra information collected from the caller and supplied in the following outputs.

II Digits

This is a value supplied in accordance with the Sprint specification and specific to a given customer account.

CRID

This is a value supplied in accordance with the Sprint specification and specific to a given customer account.

Digit Parameters Type List

This is a list of values that describe any supplied digit parameters in accordance with the Sprint specification.

Digit Parameters Nature List

This is a parallel list of values that describe any supplied digit parameters in accordance with the Sprint specification.

Digit Parameters Digits List

This is a parallel list of values that contain any supplied digit parameters in accordance with the Sprint specification.

Enterprise Group List

This is the list of Enterprise Groups that have been configured in Interaction Administrator to be queried for the dialed number supplied in the output above.

Key String

This is a sequence number that is guaranteed to be unique across all requests on a given Director server regardless of source or time received.

Request Source

This is an ID which identifies the source of the message. You must copy this to the similarly-named input on the any of the tools used to send a response back to the carrier.

Request Sequence Number

This is an ID which identifies the message. You must copy this to the similarly-named input on the any of the tools used to send a response back to the carrier.

Simulated Request

This is a Boolean value that indicates whether this request was sent from the simulator dialog.

Related Topics

 [Director Tools in Interaction Designer](#)

Accept ATT Route Request initiator

This initiator fires when a route request is received from ATT.

Initiator Properties Page

Notification Object Type

IDirector Routing

Object ID

{all} – do not choose this value or else requests from all carriers will come to the handler.

ATT – choose this value to receive only requests from ATT.

Notification Event

Send Route Request – choose this for live operation. It is the event that is fired by the Director software to indicate that a route request has been received from the specified carrier.

Simulated Route Request – choose this for testing. It is the event that is fired by the Director software when a simulated route request is sent from the simulator dialog. In this way the actual handler that will be used in a live system can be tested for a given set of inputs.

Outputs

Dialed Number

This is the number that the carrier's request indicates was dialed by the caller.

Calling Number

This is the number that the carrier's request indicates the caller is calling from.

Customer Entered Digits

This is a set of digits collected in the carrier's network if such a facility is available.

Enterprise Group List

This is the list of Enterprise Groups that have been configured in Interaction Administrator to be queried for the dialed number supplied in the output above.

Key String

This is a sequence number that is guaranteed to be unique across all requests on a given Director server regardless of source or time received.

Request Source

This is an ID which identifies the source of the message. You must copy this to the similarly-named input on the any of the tools used to send a response back to the carrier.

Request Sequence Number

This is an ID which identifies the message. You must copy this to the similarly-named input on the any of the tools used to send a response back to the carrier.

Simulated Request

This is a Boolean value that indicates whether or not this request was sent from the simulator dialog.

Related Topics

 [Director Tools in Interaction Designer](#)

Accept Generic Routing Statistics initiator

This initiator fires when Director generates a set of statistics regarding a received route request.

Initiator Properties Page

Notification Object Type

IDirector Statistics

Object ID

{all} – this is the only choice. The values in the object ID are not currently meaningful.

Notification Event

{all} – choose this to receive statistics on all types of request outcomes. The values in the object id will indicate the outcome of the request.

Routing Success

Choose this to receive statistics only on successful outcomes.

Routing Failure

Choose this to receive statistics only on failed outcomes.

Routing Timeout

Choose this to receive statistics only on timeout outcomes.

Outputs

Primary Key

This is a unique identifier for each request and could be used as a DB record key.

Correlation Id

This is an additional unique identifier.

Request Received Time

This is the time that the request was received by Director.

Response Time

This is the time in milliseconds that the request took to be processed.

IDirector Site Id

This is site ID of the Director server.

Originating Carrier

This is the carrier that sent the request.

Originating Site Id

This value is not currently meaningful.

Called Info

This is the number that the request indicated was dialed by the caller.

Calling Info

This is the number that the request indicated the caller is calling from.

Customer Entered Digits

This is a set of digits collected in by any network IVR and sent with the request.

Sprint Info Digits

This is only meaningful if the request came from Sprint and only if Sprint supplied this information in the request. The content of this value is determined by the Sprint specification.

Sprint CRID

This is only meaningful if the request came from Sprint and only if Sprint supplied this information in the request. The content of this value is determined by the Sprint specification.

Result

This is a value that indicates whether the request was responded to at all and if so whether it was successful or resulted in an error.

Response Type

If a response was sent, this indicates the type. The actual value depends on which carrier sent the request.

Response Code

If a response was sent, this indicates the code. The actual value depends on which carrier sent the request and the type of the response.

Routed Site

If a route was chosen successfully, this indicates the name, as configured in Director's IA, of the destination monitored server.

Routed Queue

If a route was chosen successfully, this indicates the name, as configured in Director's IA, of the destination monitored queue.

Routed Agent

This value is not currently meaningful.

Routed IA Profile

This value is not currently meaningful.

Additional Info

This is a list of strings that may contain extra information supplied by the Director software.

Related Topics

 [Director Tools in Interaction Designer](#)

Director Connection Lost initiator

This initiator fires when the last (or only, if there is not more than one) Director connection is lost. Connection loss handling is done by Director software. However, if you want to create a custom action, such as sending an alert after the last connection loss, you can create a handler using this initiator.

Initiator Properties

Notification Object Type

IDirector Status

Object ID

{all} – this is the only choice. The values in the object id are not currently meaningful.

Notification Event

Director Connection Lost – you must choose this. It is the event that is fired by the Director software to indicate that the last connection has been lost.

Related Topics

 [Director Tools in Interaction Designer](#)

Director Message Received initiator

This initiator fires when a message is received as a result of the DirectorSendMessage tool being executed on either the Director server or another CIC server.

Initiator Properties Page

Notification Object Type

IDirector Routing

Object ID

{all} – this is the only choice. The values in the object id are not currently meaningful.

Notification Event

Director Message Received – you must choose this. It is the event that is fired by the Director software to indicate that a message has been received.

Outputs

Message Handle

This is the handle of a UMF (Universal Message Format) message received from Director. The message may actually have come from another CIC server connected to Director. Any valid operations on a UMF message are allowed. It is up to the endpoints to know the contents of the message. A handler with an instance of this initiator already exists in the system, so if someone creates another handler, it too will receive all of the messages that the original handler receives. Ignore these messages in any custom handlers that use this initiator.

Response Correlation Id

This is an ID which identifies the message. You must copy this to the similarly-named input on the DirectorSendResponse tool.

Response Destination Id

This is an ID which identifies the source of the message. You must copy this to the similarly-named input on the DirectorSendResponse tool.

Related Topics

 [Director Tools in Interaction Designer](#)

Director Connection Restored initiator

This initiator is fired when the first (or only, if there is not more than one) Director server makes a connection to the CIC server.

Initiator Properties Page

Notification Object Type

IDirector Status

Object ID

{all} – this is the only choice. The values in the object id are not currently meaningful.

Notification Event

Director Connection Restored – you must choose this. It is the event that is fired by the Director software to indicate that a connection has been established.

Outputs

Response Correlation Id

This is an ID which identifies the message. You must copy this to the similarly-named input on the DirectorMonitoredServerReady tool.

Response Destination Id

This is an ID which identifies the source of the message. You must copy this to the similarly-named input on the DirectorMonitoredServerReady tool.

Interaction Id (String) List

This is a list of interaction IDs in string form, which represents the pending Director-handled interactions at the point of connection.

Director Monitored Workgroup Names List

This is the list of names of workgroups that the recently-connected Director server is configured to monitor as candidate destinations.

Director Enabled Queue Names List

This is the list of names of workgroups whose queues are configured to process arriving interactions through Director.

Related Topics

 [Director Tools in Interaction Designer](#)

Internal Use Only Tools

Director Get Overflow Spec

This tool is for internal use only.

 [Director Tools in Interaction Designer](#)

Director Get Time String tool

This tool is for internal use only.

 [Director Tools in Interaction Designer](#)

Director Interaction Removed tool

This tool is for internal use only.

 [Director Tools in Interaction Designer](#)

Director Interaction Transferred tool

This tool is for internal use only.

 [Director Tools in Interaction Designer](#)

Director Process Offering Interaction tool

This tool is for internal use only.

 [Director Tools in Interaction Designer](#)

Director Remote Audio Path Complete tool

This tool is for internal use only.

 [Director Tools in Interaction Designer](#)

Director Remote Routing tool

This tool is for internal use only.

 [Director Tools in Interaction Designer](#)

Director Route Email tool

This tool is for internal use only.



[Director Tools in Interaction Designer](#)

Director Send Message Tool

This Director tool sends a message to the Director server. The message may be destined for Director itself or for another monitored CIC server. The Director server identifies where the message should go using the destination ID. If the destination is a site id, Director will route it to the appropriate monitored CIC server, if connected.

Note: This is not a general purpose tool and can only be used under limited circumstances. This tool can appear on both an Interaction Director server and a CIC monitored server.

Inputs

Message Handle

This is the handle of the UMF message that is to be sent. Presumably it has been populated with values of interest to the application that will receive it. The content of the message is completely defined by the applications using it.

Destination

The only values that should be specified here are the site ID of a connected monitored server.

Timeout (ms)

If a response is not received within this amount of time, the Timeout exit path will be taken. To specify that no response is expected, this value should be set to 0.

Outputs

Response Message Handle

This is the handle of the UMF message that was received in response if a response was expected. The content of the message is completely defined by the applications using it. If no response was expected then this value is not value and should not be accessed.

Exit Paths

Success

If a response was expected then this path means that one has been received and its handle is in the output. If no response was expected, then this path means the message was successfully sent.

Failure

This path will be taken if the message could not be sent.

Timeout

All messages regardless of destination are routed through the Director server. If either the Director server or the destination server does not send a response (if one is expected) then this path will be taken

Unknown Destination

If destined for another monitored CIC server and that server is not connected to Director, then this path will be taken.

Related Topics

[Director Tools in Interaction Designer](#)

Director Send Response Tool

This Director tool sends a response message to a previous message received from the Director server. The received message may have come from Director itself or from another monitored CIC server. The Director server identifies where the response should go and what message it is for using the destination ID and correlation ID values. These must be copied from the original message.

Note: This is not a general purpose tool and can only be used under limited circumstances. This tool can appear on both an Interaction Director server and a CIC monitored server.

Inputs

Message Handle

This is a handle to a previously created UMF message that presumably has been populated with values expected in response to the original message.

Response Correlation Id

This value identifies which message the response is for and should be copied from that original message

Response Destination Id

This value identifies the sender of the original message and should be copied from there.

Outputs

None.

Exit Paths

Success

This path is taken if the response message was sent.

Failure

This path indicates that the message could not be sent.

Unknown Destination

This path is taken when the supplied destination ID is not known by the Director server

Related Topics

[Director Tools in Interaction Designer](#)

Email

Introduction to E-mail Tools

The E-mail tools are for retrieving and sending e-mail and voice mail. Refer to the *CIC ACD Processing Technical Reference* for more information about how to configure e-mail for a custom workgroup.

Note: The E-mail tools cannot access folders located on user's local machines. In order for the Open Folder tool to work, the folder being accessed must reside on the mail server. In other words, if you plan to allow users to access their voice mail messages remotely (over the telephone), then users must have their email delivered to their mailbox on the Exchange or Domino server.

Click on a tool below to learn more about that tool:

[Change Message Status](#)

[Change Message Status By Cookie](#)

[Create Folder](#)

[Delete Folder](#)

[Delete Message](#)

[Delete Message By Cookie](#)

[Email Interaction Create](#)

[Email Interaction Disconnect](#)

[Email Interaction Get Message](#)

[Email Interaction Hold](#)

[Email Interaction Insert Attachment](#)

[Email Interaction Park](#)

[Email Interaction Send Message](#)

[Email Interaction Transfer](#)

[Email Interaction Update Message](#)

[File To Recording](#)

[Find Message](#)

[Find Messages](#)

[Forward Message](#)

[Get Contained Folder](#)

[Get Cookie From Message](#)

[Get Message Count](#)

[Get Message From Cookie](#)

[Get Out of Office Status](#)

[Get Quota](#)

[Get Quota Resources](#)

[Get Quota Roots](#)

[Get Recipient Info](#)

[Is Distribution List Member](#)

[Mail Exchanger DNS Lookup](#)

[Make Attached File](#)

[Make Cookie](#)

[Make Email Body](#)

[Move Message](#)

[Move Message Ex](#)

[Open Attached File](#)

[Open Attached Message](#)

[Open Message](#)

[Open Folder](#)

[Open Message](#)

[Open Message By Cookie](#)

[Parse Cookie](#)

[Parse Email Body](#)

[Query Mail System](#)

[Reply-to Message](#)

[Send Email](#)

[Send Fax](#)

[Send Message Light Notification](#)

[Send Voice Mail](#)

[Send VPIM Message](#)

[Set Out of Office Status](#)

[Update Folder](#)

Related Topics

[Email Tool Result Codes](#)

Change Message Status

Use this Email tool to change the status of a message to either "read" or "unread."

Inputs

Folder ID

The unique identifier for the folder containing the message to be changed. This ID is retrieved with an [Incoming Mail](#) initiator, or the [Open Folder](#) tool.

Index

The zero-based index of a message within the given folder. The index denotes the location of the message within the specified folder. The value for this parameter is generated by the [Incoming Mail](#) initiator or the [Find Message](#) tool.

Status

The new status of the message. This parameter can have a value of either "read" or "unread."

Asynchronous

If set to True, the tool will exit and will not wait for the message status to change before exiting. Set this to False if you want the tool to wait until the status has been changed before exiting.

Timeout

The number of seconds the tool should wait for a return value. If no value is returned in the specified time, this tool will take the Failure exit path.

Outputs

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This path is taken if the message's status is successfully changed.

Failure

This path is taken if the operation fails.

Change Message Status By Cookie

This Email tool changes the status of a message to either "read" or "unread" by means of a cookie.

Inputs

Cookie

The file name of the cookie.

Status

The new status of the message. This parameter can have a value of either "read" or "unread."

Asynchronous

If set to True, the tool will exit and will not wait for the message status to change before exiting. Set this to False if you want the tool to wait until the status has been changed before exiting.

Timeout

The number of seconds the tool should wait for a return value. If no value is returned in the specified time, this tool will take the Failure exit path.

Outputs

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This path is taken if the message's status is successfully changed.

Failure

This path is taken if the operation fails.

Create Folder

This Email tool creates a folder on the email server.

Note: The Email tools cannot create folders located on user's local machines. This tool creates a folder on the mail server.

Inputs

Folder ID

The unique identifier for the folder being created. This ID is retrieved with an [Incoming Mail](#) initiator, or the [Open Folder](#) tool.

Folder Name

The string value that contains the name of the folder to be created.

Timeout

The number of seconds the tool will wait for a return value. This field accepts floating point numbers, so you can enter fractions of a second for more granularity. For example, 4.5 seconds (equivalent to 4500 milliseconds).

Outputs

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

If this step executes successfully, this step will take the Success exit path.

Failure

The folder could not be created. The most likely cause is the mailbox or folder name (or both) were invalid.

Delete Folder

This Email tool deletes a folder on the email server.

Note: The Email tools cannot delete folders located on user's local machines. This tool deletes a folder on the mail server.

Inputs

Folder ID

The unique identifier for the folder being deleted. This ID is retrieved with an [Incoming Mail](#) initiator, or the [Open Folder](#) tool.

Folder Name

The string value that contains the name of the folder to be deleted.

Delete Messages

A boolean value to signify deleting messages if contained within the folder targeted for deletion.

Delete Subfolders

A boolean value to signify deleting subfolders if contained within the folder targeted for deletion.

Timeout

The number of seconds the tool will wait for a return value. This field accepts floating point numbers, so you can enter fractions of a second for more granularity. For example, 4.5 seconds (equivalent to 4500 milliseconds).

Outputs

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

If this step executes successfully, this step will take the Success exit path.

Failure

The folder could not be deleted. The most likely cause is the mailbox or folder name (or both) were invalid.

Delete Message

This Email tool deletes the indexed message from the given folder. If the email platform supports the concept of a "trash" folder and the tool has not been told to expunge the message, the message will be moved to the "trash" folder. If the email platform does not support the concept of a "trash" folder or the tool has been told to expunge the message, then the message will be deleted.

Inputs

Folder ID

A value returned by the [Open Folder](#) tool.

Index

The zero-based index of a message within the given folder. The index denotes the location of the message within the specified folder. The value for this parameter is generated by the [Incoming Mail](#) initiator or the [Find Message](#) tool.

Expunge

Set this to True if you want this tool to permanently delete the message. If set to False, the tool simply moves the message from the given folder to the current mailbox's "Trash" folder.

Asynchronous

Whether or not the tool should wait until the status has been changed.

Timeout

The number of seconds the tool should wait for a return value. If no value is returned in the specified time, this tool will take the Failure exit path.

Outputs

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This step takes the Success exit path if the message was deleted successfully.

Failure

This step takes the Failure exit path if the folder or index was invalid.

Delete Message By Cookie

This Email tool deletes a message by means of a cookie. If the email platform supports the concept of a "trash" folder and the tool has not been told to expunge the message, the message will be moved to the "trash" folder. If the email platform does not support the concept of a "trash" folder or the tool has been told to expunge the message, then the message will be deleted.

Inputs

Cookie

The file name of the cookie.

Expunge

Set this to True if you want this tool to permanently delete the message. If set to False, the tool simply moves the message from the given folder to the current mailbox's "Trash" folder.

Asynchronous

If set to True, the tool will exit will not wait for the message to be deleted before exiting. Set this to False if you want the tool to wait until the message has been deleted before exiting.

Timeout

The number of seconds the tool should wait for a return value. If no value is returned in the specified time, this tool will take the Failure exit path.

Outputs

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This step takes the Success exit path if the message was deleted successfully.

Failure

This step takes the Failure exit path if the operation fails.

Email Interaction Create

This Email tool creates an email interaction in a specific queue.

Inputs

Queue

The user or workgroup queue for which the email is being created.

Direction

The direction of the interaction.

1=Incoming, 2=Outgoing

Type

1=New, 2=Reply, 3=Auto Reply, 4=Forward

Parent

The unique identifier for the interaction's parent email interaction.

Remote Name

The value entered here is used to populate the Name field on the email interaction.

Remote Address

The value entered here is used to populate the email address for the interaction.

Local Name

The name to use for the sender of the email.

Local Address

The address from which the email is sent.

Attribute Names

A list of string value containing the attribute names to add with the interaction.

Attribute Values

A list of string value (parallel to the Attribute Names list) that contains values corresponding to the attribute names.

Outputs

Interaction Id

The unique identifier for the created email interaction.

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This path is taken if the email interaction is successfully created.

Failure

This path is taken if the operation fails.

Email Interaction Disconnect

This Email tool disconnects an email interaction.

Inputs

Interaction Id

The unique identifier of the email interaction to disconnect.

User

The CIC user ID of the person on whose behalf the email is being disconnected.

Attribute Names

A list of string value containing the attribute names that are associated with the interaction to disconnect.

Attribute Values

A list of string value (parallel to the Attribute Names list) that contains values corresponding to the attribute names.

Outputs

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This path is taken if the email interaction is successfully disconnected.

Failure

This path is taken if the operation fails.

Email Interaction Get Message

This Email tool returns the email cookie representing the email message that is associated with the email interaction. Refer to the *CIC ACD Processing Technical Reference* for more information about how to configure email for a custom workgroup.

Inputs

Interaction Id

The identifier for the email interaction for which you want to get the message.

Outputs

Cookie

The value used with the [Open Message by Cookie](#) tool to get the message content.

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This path is taken if the email interaction is successfully retrieved.

Failure

This path is taken if the operation fails.

Email Interaction Hold

This Email tool lets you set or toggle the held state of the interaction.

Inputs

Interaction Id

The identifier for the email interaction for which you want to set the held state.

User

The CIC user ID of the person on whose behalf the email interaction held state is being set.

Hold State

1=Set, 2=Unset, 3=Toggle

Attribute Names

A list of string value containing the attribute names that are associated with the interaction.

Attribute Values

A list of string value (parallel to the Attribute Names list) that contains values corresponding to the attribute names.

Outputs

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This path is taken if the email interaction is successfully held.

Failure

This path is taken if the operation fails.

Email Interaction Insert Attachment

This Email tool can be used to insert a file into an email interaction.

Inputs

Interaction Id

The identifier for the email interaction to which the file should be attached.

Attached File

Data type output from another email tool, such as [Open Message](#), [Open Message by Cookie](#), or [Make Attached File](#).

Outputs

Attached File

The handle to the attached file.

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This path is taken if the email attachment is successfully inserted.

Failure

This path is taken if the operation fails.

Email Interaction Park

This Email tool lets you park an email interaction on another user's queue.

Inputs

Interaction Id

The unique identifier for the email interaction to park.

Queue

The user's queue to park the interaction on.

User

The CIC user ID of the person on whose behalf the email interaction is being parked.

Attribute Names

A list of string value containing the attribute names that are associated with the interaction to park.

Attribute Values

A list of string value (parallel to the Attribute Names list) that contains values corresponding to the attribute names.

Outputs

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This path is taken if the email interaction is successfully parked.

Failure

This path is taken if the operation fails.

Email Interaction Record

This Email tool enables or disables email recording for an interaction. The Record State input changes the recording status, while the other inputs assign attributes to the interaction.

Inputs

Interaction Id

The identifier for the email interaction to record.

User

The user name to assign to the interaction.

Record State

Set, Unset, or Toggle.

Supervisor

The supervisor name to assign to the interaction.

Attribute Names

Names of custom attributes to assign to the interaction.

Attribute Values

Values for the custom attributes specified in the Attribute Names input.

Outputs

Result

The result of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This path is taken if the email recording is successfully set or unset.

Failure

This path is taken if the operation fails.

Email Interaction Send Message

This Email tool sends the message associated with an email interaction.

Inputs

Interaction Id

The unique identifier for the email interaction.

Outputs

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This path is taken if the email message is successfully sent.

Failure

This path is taken if the operation fails.

Email Interaction Transfer

This Email tool lets you transfer an interaction to a specified user or workgroup queue. This tool generates an EmailInteractionTransferred event. This event starts a handler with the Email Interaction transferred initiator, such as the System_QueueEmailOfferingNonSystemQueue handler. Refer to the *CIC ACD Processing Technical Reference* for more information about how to configure email for a custom workgroup.

Inputs

Interaction Id

The unique identifier for the email interaction to transfer.

Queue

The user or workgroup queue to which the email is being transferred.

User

The CIC user ID of the person for whom the interaction is being transferred.

Attribute Names

A list of string value containing the attribute names that are associated with the interaction to transfer.

Attribute Values

A list of string value (parallel to the Attribute Names list) that contains values corresponding to the attribute names.

Outputs

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This path is taken if the email interaction is successfully transferred.

Failure

This path is taken if the operation fails.

Email Interaction Update Message

This Email tool lets you update the content of a message, such as an Outlook message, that is associated with an email interaction.

Inputs

Interaction Id

The unique identifier of the email interaction to update.

Sender

Optionally specifies a local email address from which this message should be sent. This input can accept a string of the email address or a string of the display name with the email address.

Examples:

bojo@xyz.com

Boris Johnson|bojo@xyz.com

The email system will generate the updated message in the outbox folder of the mailbox associated with this address, so that it will appear to the recipients as if the message originated from the owner of that mailbox. The transmission of the updated message will fail if the mailbox associated with this address is not accessible by the email system. If no From address is specified, the message will originate in the outbox folder of the mailbox belonging to the account under which the CIC email system is running. Separate multiple email addresses with a semicolon.

Subject

Specifies the text to appear on the subject line of the email message. You may enter a literal value like "Caller disconnected without leaving a voice mail message" or build a complex expression using the [Expression Editor Assistant](#).

Importance

Specifies the importance of the message as Low, Normal, or High.

Sensitivity

This parameter is a string that can have the value of None, Personal, Private, or Confidential. The default is None and this will also be used in the event that something other than one of the other three valid values is entered.

To Recipients

Specifies the recipient of the message. Separate multiple email addresses with a semicolon. Separate multiple email addresses with a semicolon. This input can accept a string of the email address or a string of the display name with the email address.

Examples:

bojo@xyz.com

bojo@xyz.com;tsmith@abc.com

Boris Johnson|bojo@xyz.com

Boris Johnson|bojo@xyz.com;tsmith@abc.com

If you leave the To and CC parameters empty, the message is sent to the address specified in the Unaddressed Mail Recipient [server parameter](#).

CC Recipients

Specifies who should receive a copy of this email. Separate multiple email addresses with a semicolon. Separate multiple email addresses with a semicolon. This input can accept a string of the email address or a string of the display name with the email address.

Examples:

bojo@xyz.com

bojo@xyz.com;tsmith@abc.com

Boris Johnson|bojo@xyz.com

Boris Johnson|bojo@xyz.com;tsmith@abc.com

BCC Recipients

Optionally specifies recipients who should receive a blind carbon copy of the updated message. Separate multiple email addresses with a semicolon. Separate multiple email addresses with a semicolon. This input can accept a string of the email address or a string of the display name with the email address.

Examples:

bojo@xyz.com

bojo@xyz.com;tsmith@abc.com

Boris Johnson|bojo@xyz.com

Boris Johnson|bojo@xyz.com;tsmith@abc.com

Reply To

Optionally specifies one or more email addresses to which the recipients of the message should direct their own replies. Separate multiple email addresses with a semicolon. Separate multiple email addresses with a semicolon. This input can accept a string of the email address or a string of the display name with the email address.

Examples:

bojo@xyz.com

bojo@xyz.com;tsmith@abc.com

Boris Johnson|bojo@xyz.com

Boris Johnson|bojo@xyz.com;tsmith@abc.com

Body

A List of Email Body type. Specifies the text to appear in the body of the email message. You can enter a literal value or build a complex expression using the [Expression Editor Assistant](#).

Delivery Receipt

Set to True to prompt the recipient for a delivery receipt upon receiving the email. If the underlying mail system does not support this feature, this value will be ignored.

Read Receipt

Set to True to prompt the recipient for a read receipt upon opening the email message. If the underlying mail system does not support this feature, this value will be ignored.

Message Class

The class of the message (text, voice, fax, or ndr). The tool will only search messages of the specified class.

Outputs

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This path is taken if the email message is successfully updated.

Failure

This path is taken if the operation fails.

File To Recording

This Email tool converts a file into a recording. The recording ID of the object is passed to various tool steps for processing.

Inputs

Filename

The full path and file name of the file to be converted into a recording ID.

Delete File

Set to True if you want the file to be deleted after this tool exits.

Outputs

Recording

The Recording ID of the converted file.

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This path is taken if the file is successfully converted.

Failure

This path is taken if the operation fails.

Find Message

This Email tool searches the specified folder for the first message matching the given criteria. Any empty parameters will not be included in the search.

Inputs

Folder ID

The unique identifier for the folder that will be searched. This ID is retrieved with an [Incoming Mail](#) initiator, or the [Open Folder](#) tool.

Index

The index of the various messages in each folder indicates the order in which they will be searched (i.e., first, second, third, etc.) This parameter is the number of messages that is skipped before the search begins. When searching forward, zero is the index used to find the first message (i.e., zero messages are skipped). After the first search, the index of the previous matching message plus one will be used.

Example: The first matching message in the folder is third in the index. When first searching this folder, the Index parameter is zero, and so no messages are skipped. After finding the first match (at index three), this tool is called again, this time with an Index parameter of 4.

Search Backwards

Select this option to search messages in the folder in reverse, beginning with the last message. The search starts with the specified index and can be used to find the most recent message first.

Status

Messages can have a status of "read", indicating that the message has already been read, or "unread", indicating that the message has not been read. Only messages of the selected status will be searched.

Importance

The Importance of a message is "low", "high", or "normal". This tool will only search messages with the specified Importance.

Sensitivity

The Sensitivity of a message can be "personal", "private", "confidential", or "none". This tool will only search messages with the specified Sensitivity.

Message Class

The Message Class of a message can be "text", "voice", "fax", or "ndr". This tool will only search messages with the specified Message Class.

Subject

User regular expressions to search for messages based on their field.

Sender

Use regular expressions to search on the friendly name or email address of a particular sender.

Start Date/Time

Only message received after this date and time will be searched.

End Date/Time

Only messages received before this date and time will be searched.

Timeout

The number of seconds the tool should wait for a return value. If no value is returned in the specified time, this tool will take the Failure exit path.

Outputs

Index

The number indicating the position of the first matching message in the folder.

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Found

The Found exit path is taken if a message matching the search criteria was found.

Not Found

The Not Found path is taken if no matching message was found.

Failure

This path is taken if the operation fails.

Find Messages

This Email tool is identical to the [Find Message](#) tool except that it will return all matching messages instead of just the first match.

Inputs

Folder ID

The unique identifier for the folder containing the message to be changed. This ID is retrieved with an [Incoming Mail](#) initiator, or the [Open Folder](#) tool.

Status

Only messages with this status will be searched, can be "read" or "unread".

Importance

Only messages with this importance will be searched, can be "low", "high", or "normal".

Sensitivity

Only messages with this sensitivity will be searched, can be "personal", "private", "confidential", or "none".

Message Class

Only messages with this class will be searched, can be "text", "voice", "fax", or "ndr".

Subject

Only messages with this subject will be searched. Uses a regular expression.

Sender

Only messages from this sender will be searched (friendly name or email address). Uses a regular expression.

Start Date/Time

Only message received after this date and time will be searched.

End Date/Time

Only messages received before this date and time will be searched.

Timeout

The number of seconds the tool should wait for a return value. If no value is returned in the specified time, this tool will take the Failure exit path.

Outputs

Indices

The list of indices of the matching messages.

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Found

The Found exit path is taken if at least one message matching the search criteria was found.

Not Found

The Not Found path is taken if no matching message was found (so none of the output values are valid).

Failure

This path is taken if the operation fails.

Forward Message

This Email tool forwards an e-mail (or voice or fax mail) to another email user. If a recording is provided, the resulting message will be a voicemail unless another message class is given. This tool (unlike the [Send Email](#), [Send Fax Mail](#), and [Send Voice Mail](#) tools) performs some synchronous interaction with the mail server before queuing messages for alter sending.

Inputs

Folder ID

The folder that contains the email to be forwarded. This ID is retrieved with an [Incoming Mail](#) initiator, or the [Open Folder](#) tool.

Index

The zero-based index of the message within the given folder. The value for this parameter is generated by the [Incoming Mail](#) initiator or the [Find Message](#) tool.

To Recipients

Specifies the recipient of the message. Separate multiple email addresses with a semicolon. If you leave the To and CC parameters empty, the message is sent to the address specified in the Unaddressed Mail Recipient [server parameter](#).

CC Recipients

Specifies who should receive a copy of this email. Separate multiple email addresses with a semicolon.

BCC Recipients

Optionally specifies recipients who should receive a "Blind Carbon Copy" of the reply message. Separate multiple email addresses with a semicolon.

Reply To

Optionally specifies one or more email addresses to which the recipients of the reply message should direct their own replies. Separate multiple email addresses with a semicolon.

Sender

Optionally specifies a local email address from which this reply message should be sent. The email system will generate the reply message in the outbox folder of the mailbox associated with this address, so that it will appear to the recipients as if the message originated from the owner of that mailbox. The transmission of the reply message will fail if the mailbox associated with this address is not accessible by the email system. If no From address is specified, the message will originate in the outbox folder of the mailbox belonging to the account under which the CIC email system is running. Separate multiple email addresses with a semicolon.

Subject

Specifies the text on the subject line of the email message. This value is taken from the message that is forwarded.

Importance

Specifies the importance of the message as "Low," "Normal," or "High."

Sensitivity

This parameter is a string that can have the value of "None", "Personal", "Private", or "Confidential". The default is "None", and this will also be used in the event that something other than one of the other three valid values is entered.

Body

A List of Email Body type. Specifies the text to appear in the body of the email message. You may enter a literal value or build a complex expression using the [Expression Editor Assistant](#).

Message Attachments

An optional variable that identifies a message to be attached to the reply message.

Attached Files

A list of strings containing zero or more complete file system paths to files that will be copied into the reply message. As with the [Send Email](#) tool, this parameter is a *list* of strings, not a single string, so you cannot just type the file path in this parameter. Instead you must assign the attachment file paths to the elements in variable of type list of string.

Note: OLE attachments (for example, bitmaps rendered inline) are not visible, as attachments or otherwise, to handlers. As far as the handler tools are concerned, an email message consists of a text body and zero or more **file** attachments. Other sorts of attachments, such as embedded OLE objects and "nested" messages, are not accessible through the handler tools, since they are not supported or easily simulated on messaging systems other than Exchange (i.e. IBM Notes or IMAP).

Delete Attachments

When checked, all files in the Attachments List are deleted after they have been attached (copied) to the email message and successfully sent. Sometimes, especially when replying to or forwarding a message, you should create a temporary file and attach it to the message being sent. After the message is sent the temporary file needs to be deleted, but since outgoing mail is handled asynchronously in IP, the handler can't delete the file immediately after the tool returns, because the message most likely hasn't yet been processed. This checkbox simply passes the responsibility for deleting the file(s) on to the asynchronous service thread that actually sends the message since that thread is the only one that actually knows when it's safe to delete it (or them).

Recording ID

The file name of an attached audio recording.

Audio Format

The format to use for audio recording. See [Compress Audio File](#) tool for possible values.

Normalize

Set this Boolean to True to normalize the audio recording. When a recording is normalized, it is analyzed to determine what the maximum volume level of the audio file is. A value 5% below the maximum value is then used to set the gain value that will bring the maximum volume level up or down to a standard level. This ensures that at a given station, all recordings will play back at the same relative volume. Using a value 5% below the maximum volume to calculate the gain prevents a short burst of static or similar anomalous noise from throwing off this volume adjustment.

Delivery Receipt

Set to True to prompt the recipient for a delivery receipt upon receiving this email. If the underlying mail system does not support this feature, it will be ignored.

Read Receipt

Set to True to prompt the recipient for a read receipt upon opening this email. If the underlying mail system does not support this feature, it will be ignored.

Saved Copies

String designating the mailbox cookie in which copies of the email are saved.

Message Class

The class of the message. Can be "text," "voice," and "fax."

Timeout

The number of seconds that the tool will wait for a return value.

Scheduled

The DateTime variable for the future delivery of a message.

Attachment Files

A list of attached file objects output from another tool, such as [Make Attached File](#).

Outputs

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This tool (unlike the Send Email, Send Fax Mail, and Send Voice Mail tools) performs some synchronous interaction with the mail server before queuing messages for later sending. It will take the Success exit path if the message was queued for sending, but this does not guarantee that the message was sent or that the recipient's email address is valid.

Failure

This step takes the Failure exit path if the connection to mail server is dropped, but this is rare because such a failure would cause one of the previously called email tools to fail.

Get Contained Folders

This Email tool queries a folder for a list of the names of all of the sub-folders contained within it.

Inputs

Folder ID

The unique identifier for the folder that will be queried for sub-folders. This ID is retrieved with an [Incoming Mail](#) initiator, or the [Open Folder](#) tool.

Timeout

The number of seconds the tool should wait for a return value. If no value is returned in the specified time, this tool will take the Failure exit path.

Outputs

Folder Names

A list of strings naming the sub-folders contained within the given folder.

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This step takes the Success exit path if the folder was successfully queried. When the given folder has no sub-folders, the Names list is empty and the Success path is still taken.

Failure

This step takes the Failure exit path if the email server is not accessible for any reason.

Get Cookie From Message

This Email tool creates a string (i.e. cookie) from a Folder ID and an Index. This cookie can be used later by other handlers to retrieve the message.

The cookie is only valid as long as the message remains in the same folder it was in when the cookie was created.

Inputs

Folder ID

The unique identifier for the folder containing the message. This ID is retrieved with an [Incoming Mail](#) initiator, or the [Open Folder](#) tool.

Index

The zero-based index of the message within the given folder. The value for this parameter is generated by the [Incoming Mail initiator](#) or the [Find Message](#) tool.

Timeout

The number of seconds the tool will wait for a return value.

Outputs

Cookie

The string generated to be used later to retrieve the message.

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This path is taken if the cookie is successfully generated.

Failure

This path is taken if the operation fails.

Get Message Count

This Email tool returns the number of messages in a folder that match the given criteria.

Inputs

Folder ID

The unique identifier for the folder being queried. This ID is retrieved with an [Incoming Mail](#) initiator, or the [Open Folder](#) tool.

Status

The status the message should be changed to. This parameter can have a value of either "read" or "unread."

Importance

Specifies the importance of the message as "Low," "Normal," or "High."

Sensitivity

This parameter is a string that can have the value of "None", "Personal", "Private", or "Confidential". The default is "None", and this will also be used in the event that something other than one of the other three valid values is entered.

Message Class

Only messages with this class will be searched, can be "text," "voice," "fax," or "ndr."

Subject

Only messages with this subject will be searched. Uses a regular expression.

Sender

Only messages from this sender will be searched (friendly name or email address). Uses a regular expression.

Start Date/Time

Only message received after this date and time will be searched.

End Date/Time

Only messages received before this date and time will be searched.

Timeout

The number of seconds the tool should wait for a return value. If no value is returned in the specified time, this tool will take the Failure exit path.

Outputs

Unread Count

The number of unread messages that match.

Read Count

The number of read messages that match.

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This path is taken if the message count is successfully retrieved.

Failure

This path is taken if the operation fails.

Get Message From Cookie

This Email tool uses the cookie generated by the [Get Cookie From Message tool](#) and returns the Folder ID and Index of the message represented by the cookie.

Inputs

Cookie

The string generated to be used later to retrieve the message.

Timeout

The number of seconds the tool will wait for a return value.

Outputs

Folder ID

The folder value returned by the [Open Folder](#) tool.

Index

The zero-based index of the message within the given folder. The value for this parameter is generated by the [Incoming Mail initiator](#) or the [Find Message](#) tool.

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This path is taken if the message is successfully retrieved.

Failure

This path is taken if the operation fails.

Get Out of Office Status

This Email tool returns the Out Of Office (OOO) status for the given Exchange mailbox. This feature is can only be used with CIC systems that are connected to an Exchange mail server.

On Notes or Groupwise, this tool will fail and will create an appropriate entry in the Application Event Log.

Note: The handler author must have intimate knowledge of the mail system in use in order to effectively use this tool, as all parameters can change from one application to the next.

Inputs

Mailbox

String representing the mailbox cookie for the mailbox.

Outputs

Out Of Office

A boolean set to "true" is an OOO status has been set.

Out of Office Text

A string that contains the text of the OOO status.

Result

This integer is for advanced diagnostic purposes and is for use by PureConnect Customer Care.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This path is taken if the OOO status is successfully retrieved.

Failure

This path is taken if the operation fails. The most common reason for failure is a lack of OOO support by the mail system.

Get Quota

This Email tool returns the implementation specific quota for the given folder, root and resource.

This tool is currently only supported by IMAP. On Notes or Exchange, this tool will fail and will create an appropriate entry in the Application Event Log.

Note: The handler author must have intimate knowledge of the mail system in use in order to effectively use this tool, as all parameters can change from one application to the next. Usage and limit, as well as the number of roots and/or resources per root can also vary from one mail system to another.

Inputs

Folder ID

The unique identifier for the folder being queried. This ID is retrieved with an [Incoming Mail](#) initiator, or the [Open Folder](#) tool.

Root Name

A string representing the root name (e.g., "user/troyt").

Resource Name

A string representing the resource name (e.g., "storage").

Timeout

The number of seconds the tool will wait for a return value.

Outputs

Limit

Integer representing the limit. This number could represent a number of messages or kilobytes, or whatever else the system uses to determine its limits.

Usage

Integer representing the current usage. This number could represent a number of messages or kilobytes, or whatever else the system uses to determine usage parameters.

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This path is taken if the quota is successfully retrieved.

Failure

This path is taken if the operation fails. The most common reason for failure is a lack of quota support by the mail system.

Get Quota Resources

This Email tool returns the resources associated with the given folder and root.

This tool is currently only supported by IMAP. On Notes or Exchange, this tool will fail and will create an appropriate entry in the Application Event Log.

Note: The handler author must have intimate knowledge of the mail system in use in order to effectively use this tool, as all parameters can change from one application to the next.

Inputs

Folder ID

The unique identifier for the folder being queried. This ID is retrieved with an [Incoming Mail](#) initiator, or the [Open Folder](#) tool.

Root Name

A string representing the root name (e.g., "user/troyt").

Timeout

The number of seconds the tool will wait for a return value.

Output

Quota Resources

List of strings representing all of the resources for the given root.

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

The path is taken if the quota resources are successfully retrieved.

Failure

This path is taken if the operation fails. The most common reason for failure is a lack of quota support by the mail system.

Get Quota Roots

This Email tool returns the roots associated with the given folder.

This tool is currently only supported by IMAP. On Notes or Exchange, this tool will fail and will create an appropriate entry in the Application Event Log.

Note: The handler author must have intimate knowledge of the mail system in use in order to effectively use this tool, as all parameters can change from one application to the next.

Input

Folder ID

The unique identifier for the folder being queried. This ID is retrieved with an [Incoming Mail](#) initiator, or the [Open Folder](#) tool.

Timeout

The number of seconds the tool will wait for a return value.

Output

Quota Roots

List of strings representing all of the root names.

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This path is taken if the quota roots are successfully retrieved.

Failure

This path is taken if the operation fails. The most common reason for failure is a lack of quota support by the mail system.

Get Recipient Info

This Email tool provides the friendly name and email address of a recipient.

Inputs

Recipient

The email recipient

Outputs

Friendly Name

The friendly name of the email recipient.

Email Address

The email address of the recipient.

Exit Paths

Success

This path is taken if the information is successfully retrieved.

Failure

This path is taken if the operation fails.

Is Distribution List Member

This Email tool returns an indicator as to whether or not an e-mail address or moniker is part of an existing distribution list.

Inputs

One of the following:

Email Address

The e-mail address used to determine whether or not the address is a member of a distribution list.

Moniker

The string moniker for the distribution list.

Outputs

Result

This integer is for diagnostic purposes and is for use by PureConnect Customer Care.

See [Email Tool Result Codes](#) for more information.

Exit Paths

True

The e-mail address or moniker is a distribution list member.

False

The e-mail address or moniker is not a distribution list member.

Failure

The tool failed. Consult the result for more information.

Mail Exchanger DNS Lookup

Mail Exchangers (or MX records) are part of the DNS information for a domain. The MX record is an ordered list of destinations that tells mailers where to send messages if they want to a given domain. This Email tool looks up the Mx record of an email address.

Inputs

Mailbox

The email domain being looked up.

Outputs

Mail Exchanger

The Mx record of the specified domain.

Exit Paths

Success

This path is taken if the Mx record is successfully retrieved.

Failure

This path is taken if the operation fails.

Make Attached File

This Email tool is used to create an Attached File object that can be passed as an input to another tool, such as Send E-mail. By passing an Attached File object, rather than a file path, the object can capture additional information such as a display name for the attached file, and a content identifier. This information can be used to display inline attachments in HTML message bodies.

Inputs

File

The file path.

Name

The display name.

Content ID

The content's unique identifier, which is used to display inline attachments.

Outputs

Attached File

The resulting file attachment.

Note: The tool does not assume ownership of the file in the file path. The file needs to be deleted either manually after the object is used, or automatically through the use of the Delete Attachments input on the tools that accept that data type as an input.

Exit Paths

Success

This path is taken if the attached file is successfully created.

Failure

This path is taken if the file path is not valid or the attached file cannot be created for another reason.

Make Cookie

This Email tool converts a string that is a moniker for a message into an email cookie. The moniker is a known format, however this format will vary from platform to platform. This tool performs the reverse function of the [Parse Cookie](#) tool.

Inputs

Moniker

The string output version of the cookie.

Outputs

Cookie

The email cookie to be parsed.

Exit Paths

Success

This path is taken if the cookie is successfully parsed.

Failure

This path is taken if the operation fails.

Make Email Body

This Email tool takes a content type and data and returns an Email Body type that can be passed to the message delivery tools.

Inputs

Content-type

Text/plain or text/html.

Data

The data in the format indicated by Content-type.

Outputs

Email Body

An Email Body type representing the data in the format indicated by the Content Type.

Exit Paths

Success

This path is taken if the operation is successful.

Failure

This path is taken if the operation fails.

Move Message

This Email tool moves (or copies) an indexed message from the source folder and places it in the destination folder.

Inputs

Source folder

The unique identifier for the folder currently containing the message. This ID is retrieved with an [Incoming Mail](#) initiator, or the [Open Folder](#) tool.

Index

The zero-based index of a message within the given folder. The index denotes the location of the message within the specified folder. The value for this parameter is generated by the [Incoming Mail](#) initiator or the [Find Message](#) tool.

Destination Folder

The unique identifier for the folder that will receive the message. This ID is retrieved with an [Incoming Mail](#) initiator, or the [Open Folder](#) tool.

Copy

Set to True if you want to copy the message to the destination folder instead of moving.

Asynchronous

Set to True if you want the message to be moved asynchronously. This will allow the tool to exit without waiting for the message to be moved or copied. Set to False if you want the tool to wait until the move or copy is complete before exiting.

Timeout

The number of seconds the tool should wait for a return value. If no value is returned in the specified time, this tool will take the Failure exit path.

Outputs

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This step takes the Success exit path if the message was deleted moved or copied

Failure

This step takes the Failure exit path if the server lacks sufficient rights to write into the destination folder.

Move Message Ex

This Email tool moves (or copies) an indexed message from the source folder and places it in the destination folder and provides a cookie for the new message.

Inputs

Source folder

The unique identifier for the folder currently containing the message. This ID is retrieved with an [Incoming Mail](#) initiator, or the [Open Folder](#) tool.

Index

The zero-based index of a message within the given folder. The index denotes the location of the message within the specified folder. The value for this parameter is generated by the [Incoming Mail](#) initiator or the [Find Message](#) tool.

Destination Folder

The unique identifier for the folder that will receive the message. This ID is retrieved with an [Incoming Mail](#) initiator, or the [Open Folder](#) tool.

Copy

Set to True if you want to copy the message to the destination folder instead of moving.

Timeout

The number of seconds the tool should wait for a return value. If no value is returned in the specified time, this tool will take the Failure exit path.

Outputs

Cookie

The new cookie for the message that has been moved.

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This step takes the Success exit path if the message was deleted moved or copied

Failure

This step takes the Failure exit path if the server lacks sufficient rights to write into the destination folder.

Open Attached File

This Email tool extracts a file attached to a message that has been opened with an [Open Message](#) or [Open Message By Cookie](#) tool into a specified location.

Inputs

Attached File

The Attached File to open. This comes from the Attached Files output of the Open Message or Open Message By Cookie tool.

Attachment Directory

The directory into which the file will be saved. "TEMP" will be interpreted to mean the system-defined temporary directory.

Timeout

The number of seconds the tool will wait to extract the file. If the tool times out before the file is extracted, the tool will exit via the Failure exit path with a return value indicating a timeout occurred.

Outputs

Attachment Name

The name of the extracted file as it appeared in the message.

Attachment File

The name of the file as it was saved to the file system.

Result

The return value of the operation.

See [Email Tool Result Codes](#) for more information.

Content ID

The value representing the content identifier, which is traditionally associated with an inline attachment.

Exit Paths

Success

This path is taken if the file was successfully saved.

Failure

This path is taken if the operation fails.

Open Attached Message

This email tool opens a message attached to a message that has been opened with an [Open Message](#) or [Open Message By Cookie](#) tool. The message is saved as a cookie that can then be accessed with the Open Message By Cookie tool.

Inputs

Attached Message

The Attached Message to open. This comes from the Attached Messages output of the Open Message or Open Message By Cookie tool.

Outputs

Subject

The subject of the extracted message.

Cookie

A cookie which can be used to access the entire message using the Open Message By Cookie tool.

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This path is taken if the attached message is successfully opened.

Failure

This path is taken if the operation fails.

Open Folder

This Email tool opens a folder and loads it with messages that match the given criteria. If the folder is opened asynchronously, then no message count will be available.

Note: The Email tools cannot access folders located on user's local machines. In order for the Open Folder tool to work, the folder being accessed must reside on the mail server.

Inputs

Mailbox

A mailbox identification string based on the Interaction Message Store moniker for the mailbox. The following example uses Exchange Web Services (EWS):

```
ININ.Mail.ObjectMoniker: <x-inin-mail.ex.ews.store:/s=mailbox@test.com>
```

In this example, replace mailbox@test.com with the actual SMTP address of the mailbox.

Folder Name

The string value that contains the name of the folder to be opened. Subfolders are specified by using a semicolon separator. For example, a subfolder of Inbox named "Personal" would be passed in as "Inbox;Personal." If the empty string is specified, the "Root" special folder is opened.

There are certain "special folders" that represent well-known mail folders whose names and presence are user or platform dependent.

For example:

- On an Exchange server, the special folder "Inbox" will be named "Inbox" for an English user, but not for other languages.
- The special folder "Deleted Items" might be called "Deleted Items" on an Exchange server, but "Trash" on an IMAP server.
- The special folder "Outbox" will exist on an Exchange server, but typically will not exist on an IMAP server (since IMAP is not a protocol for delivering messages).

This tool interprets these "special folders" as reserved names representing the well-known folders. This is useful, for example, if some users are on Exchange and some are on IMAP, and you want to target the correct folder for both without having to worry about whether to pass in "Deleted Items" or "Trash" for a certain user.

The special folders supported by the tool are:

- Root
- Inbox
- Outbox
- Sent Items
- Deleted Items
- Scratch - (the "Drafts" folder on Exchange)

You can use the [Query Mail System](#) tool to determine support for special folders for a particular platform.

Max Count

The maximum number of messages to open. No value or a value of 0 will retrieve all messages.

If this parameter is defined, it will be applied before any of the other filters, except when the "Message Class" parameter is used with an Exchange mailbox. In this case those two filters will be applied concurrently.

Reverse Order

Set to True to reverse the default order in which the messages are opened by the mail server. The default order is dependent on the underlying mail system.

Reverse Index

Set to True to reverse the order in which messages are indexed.

Status

Only messages with this status (read or unread) will be searched.

Importance

Only messages with this importance (low, high, or normal) will be searched.

Sensitivity

Only messages with this sensitivity (personal, private, confidential, or none) will be searched.

Message Class

Only messages with this class (text, voice, fax, or ndr) will be searched.

Subject

Only messages with this subject will be searched. Uses a regular expression.

Sender

Only messages from this sender will be searched (friendly name or email address). Uses a regular expression.

Start Date/Time

Only message received after this date and time will be searched.

End Date/Time

Only messages received before this date and time will be searched.

Asynchronous

Set this to true if you want the tool should to until the messages have been loaded before exiting.

Timeout

The number of seconds the tool will wait for a return value. This field accepts floating point numbers, so you can enter fractions of a second for more granularity. For example, 4.5 seconds (equivalent to 4500 milliseconds).

Outputs

Folder ID

An [extended type value](#) that identifies the open folder.

Unread Count

The total number of unread messages in the folder.

Read Count

The total number of read messages in the folder.

Unread Voice

The total number of unread voice mail messages in the folder.

Read Voice

The total number of read voice mail messages in the folder.

Unread Fax

The total number of unread faxes in the folder.

Read Fax

The total number of read faxes in the folder.

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

If this step executes successfully, this step will take the Success exit path.

Failure

The step takes the Failure exit path if the folder could not be opened, probably because either the mailbox or folder name (or both) was invalid.

Open Message

This Email tool extracts all of the data from an indexed message.

Inputs

Folder ID

The unique identifier for the folder containing the message. This ID is retrieved with an [Incoming Mail](#) initiator, or the [Open Folder](#) tool.

Index

The zero-based index of a message within the given folder. The index denotes the location of the message within the specified folder. The value for this parameter is generated by the [Incoming Mail](#) initiator or the [Find Message](#) tool.

Timeout

The number of seconds the tool should wait for a return value. If no value is returned in the specified time, this tool will take the Failure exit path.

Outputs

Status

The status of the message, can be "read" or "unread."

Sender

The sender of the message.

Subject

The subject of the message.

Importance

The importance (low, high, or normal) of the message.

Sensitivity

The sensitivity (private, confidential, or none) of the message.

Date/Time Received

The date and time (in UTC) the message was received.

To Recipients

The display addresses of the message's primary recipients.

CC Recipients

The display addresses of the message's secondary recipients.

Reply To

The recipients to whom replies should be sent.

Body

A List of Email Body type. The body text of the message. This contains the entire body, not just the body of the last reply.

Attachment Files

The full path names of the extracted attachment files. Attachments that are not files (i.e., Links or OLE objects) are represented by empty strings in this list.

Notes: Once an attachment has been copied into a file, the file is owned by the handler. The **Open Message** tool will not delete the file or clean it up in any way.

Note: OLE attachments (e.g. bitmaps rendered inline) are not visible, as attachments or otherwise, to handlers. As far as the handler tools are concerned, and email message consists of a text body and zero or more *file* attachments. Other sorts of attachments, such as embedded OLE objects and "nested" messages, are not accessible through the handler tools, since they are not supported or easily simulated on messaging systems other than Exchange (i.e. IBM Notes or IMAP).

Attached Messages

The messages attached to the message.

Message Class

The class (text, voice, fax, or ndr) of the message.

Cookie

The cookie of the message.

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

If this step executes successfully, this step takes the Success exit path.

Failure

If this step does not execute successfully, this step takes the Failure exit path.

Open Message By Cookie

This Email tool opens a message via a cookie.

Inputs

Cookie

The email cookie of the message.

Timeout

The number of seconds the tool should wait for a return value. If no value is returned in the specified time, this tool will take the Failure exit path.

Outputs

Status

The status of the message, can be "read" or "unread."

Sender

The sender of the message.

Subject

The subject of the message.

Importance

The importance (low, high, or normal) of the message.

Sensitivity

The sensitivity (personal, private, confidential, or none) of the message.

Date/Time Received

The time (in UTC) the message was received.

To Recipients

The display addresses of the message's primary recipients.

CC Recipients

The display addresses of the message's secondary recipients.

Reply To

The recipients to whom replies should be sent.

Body

A List of Email body type. The body text of the message. This contains the entire body, not just the body of the last reply.

Attachment Files

The full path names of the extracted attachment files. Attachments that are not files (i.e., Links or OLE objects) are represented by empty strings in this list.

Attached Messages

The messages attached to the message.

Message Class

The class (text, voice, fax, or ndr) of the message.

Result

The results (return value) of the operation.

See [Email Tool Result Codes](#) for more information.

BCC Recipients

Optionally specifies recipients who should receive a "Blind Carbon Copy" of the reply message. This tool accepts a string for single recipients and a list of strings for multiple recipients.

Exit Paths

Success

If this step executes successfully, this step takes the Success exit path.

Failure

This path is taken if the operation fails.

Parse Cookie

This Email tool converts an email cookie into a string that is a moniker for the message. The moniker is a known format, however this format will vary from platform to platform. This tool performs the reverse function of the [Make Cookie](#) tool.

Inputs

Cookie

The email cookie to be parsed.

Outputs

Moniker

The string output version of the cookie.

Exit Paths

Success

This path is taken if the cookie is successfully parsed.

Failure

This path is taken if the operation fails.

Parse Email Body

This Email tool takes an Email Body type and returns the content type and data it represents.

Inputs

Email Body

An Email Body type representing the data in the format indicated by the Content Type.

Outputs

Content-type

The Content-type associated with the data.

Data

The data in the format indicated by Content-type.

Exit Paths

Success

This path is taken if the email body is successfully parsed.

Failure

This path is taken if the operation fails.

Query Mail System

This Email tool queries the given mail system to check for support for the given "special" folders.

Input

Mailbox

String representing the mailbox cookie for the mailbox.

Timeout

The number of seconds the tool will wait for a return value.

Outputs

Supports Inbox

A Boolean variable indicating whether or not the mail system supports an Inbox folder.

Supports Outbox

A Boolean variable indicating whether or not the mail system supports an Outbox folder.

Supports Sent Items

A Boolean variable indicating whether or not the mail system supports a Sent Items folder.

Supports Deleted Items

A Boolean variable indicating whether or not the mail system supports a Deleted Items folder.

Supports Out Of Office

A boolean variable indicating whether or not the mail system supports Out Of Office messages.

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This path is taken if the query is successful.

Failure

This path is taken if the operation fails.

Reply To Message

This Email tool sends a reply message to an email (or voice or fax mail). The address of the primary recipient is taken directly from

the specified message, since the From output of the [Open Message](#) tool is a display name rather than an address and therefore may not be unique within the email system. This tool (unlike the [Send Email](#), [Send Fax Mail](#), and [Send Voice Mail](#) tools) performs some synchronous interaction with the mail server before queuing messages for later sending.

Inputs

Folder ID

The folder that contains the original message. This ID is retrieved with an [Incoming Mail initiator](#), or the [Open Folder](#) tool.

Index

The zero-based index of a message within the given folder. The index denotes the location of the message within the specified folder. The value for this parameter is generated by the [Incoming Mail](#) initiator or the [Find Message](#) tool.

Reply to All

When set to True, the reply message is addressed to all recipients of the original message. Otherwise the reply is addressed only to the sender of the original message.

To Recipients

Specifies the recipient of the message. Separate multiple email addresses with a semicolon. If you leave the To and CC parameters empty, the message is sent to the address specified in the Unaddressed Mail Recipient [server parameter](#).

CC Recipients

Optionally specifies additional recipients who should receive a copy of the reply message. Separate multiple email addresses with a semicolon.

BCC Recipients

Optionally specifies recipients who should receive a "Blind Carbon Copy" of the reply message. Separate multiple email addresses with a semicolon.

Reply To

Optionally specifies one or more email addresses to which the recipients of the reply message should direct their own replies. Separate multiple email addresses with a semicolon.

Sender

Optionally specifies a local email address from which this reply message should be sent. The email system will generate the reply message in the outbox folder of the mailbox associated with this address, so that it will appear to the recipients as if the message originated from the owner of that mailbox. The transmission of the reply message will fail if the mailbox associated with this address is not accessible by the email system. If no From address is specified, the message will originate in the outbox folder of the mailbox belonging to the account under which the CIC email system is running. Separate multiple email addresses with a semicolon.

Subject

Specifies the text on the subject line of the email message. This value is taken from the message that is being replied to.

Importance

Specifies the importance of the message as either "Low", "Normal", or "High".

Sensitivity

This parameter is a string that can have the value of "Normal", "Personal", "Private", or "Confidential". The default is "Normal", and this will also be used in the event that something other than one of the other three valid values is entered.

Include Original

Set this parameter to True to include the text of the original message in the body of the reply message. The original message will be marked as a quotation in the manner appropriate for the underlying email system.

Body

A List of Body type. Specifies the text to appear in the body of the email message. You may enter a literal value or build a complex expression using the [Expression Editor Assistant](#).

Message Attachments

A list of strings containing attached messages.

Attachment Files

A list of strings containing zero or more complete file system paths to files that will be copied into the reply message. As with the [Send Email](#) tool, this parameter is a list of strings, not a single string, so you cannot just type the file path in this parameter. Instead you must assign the attachment file paths to the elements in a variable of type "list of string."

Delete Attachments

Set this parameter to True if you want all files in the Attachments List to be deleted after they have been attached (copied) to the email message and successfully sent. It is sometimes the case, especially when replying to or forwarding a message, that you need to create a temporary file and attach it to the message being sent. After the message has been sent the temporary file needs to be deleted, but since outgoing mail is handled asynchronously in IP, the handler can't delete the file immediately after the tool returns, because the message most likely hasn't yet been processed.

Recording ID

The complete path and file name of an audio recording to be attached to the email.

Audio Format

The format to use for the audio recording. See the [Compress Audio File](#) tool for possible values.

Normalize

Set this Boolean to True to normalize the audio recording. When a recording is normalized, it is analyzed to determine what the maximum volume level of the audio file is. A value 5% below the maximum value is then used to set the gain value that will bring the maximum volume level up or down to a standard level. This ensures that at a given station, all recordings will play back at the same relative volume. Using a value 5% below the maximum volume to calculate the gain prevents a short burst of static or similar anomalous noise from throwing off this volume adjustment.

Delivery Receipt

Checking this box will prompt the recipient for a delivery receipt upon receiving this email. The underlying mail system must support this feature, otherwise it will be ignored.

Read Receipt

Checking this box will prompt the recipient for a read receipt upon opening this email. The underlying mail system must support this feature, otherwise it will be ignored.

Saved Copy

String designating the mailbox cookie in which copies of the email are saved.

Message Class

The MAPI message class for the message. The message class may be one of "Text", "Voice" or "Fax".

Timeout

The number of seconds the tool will wait for a return value.

Scheduled

The DateTime variable for the future delivery of a message.

Attachment Files

Outputs

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This tool (unlike the Send Email, Send Fax Mail, and Send Voice Mail tools) performs some synchronous interaction with the email system before queuing messages for later transmission. It will take the Success exit path if the message was queued for sending, but this does not guarantee that the message was sent or that the recipient's email address is valid.

Failure

This tool takes the Failure exit path if the connection to the email server is interrupted, but this is a rare occurrence since such an interruption would typically cause one of the other email tools to fail, such as the Open Folder or Find Message tools from which the Folder ID and Index parameters most likely originated.

Send E-Mail

This Email tool asynchronously sends an email to one or more recipients. Because this tool sends a message asynchronously, it only queues a message for sending and does not wait to verify that the message was actually sent or if the recipient's address is correct. As a result, this tool takes the Success exit path, even if the message is not sent.

Inputs

To Recipients

Specifies the recipient of the email. Separate multiple email addresses with a semicolon. This input can accept a string of the email address or a string of the display name with the email address.

Examples:

bojo@xyz.com

bojo@xyz.com;tsmith@abc.com

Boris Johnson|bojo@xyz.com

Boris Johnson|bojo@xyz.com;tsmith@abc.com

If you leave the To and CC parameters empty, the message is sent to the address specified in the Unaddressed Mail Recipient [server parameter](#).

Note: In Directory Services, each user's email address is stored in that user's User key in the mailbox attribute.

CC Recipients

Specifies who should receive a copy of this email. This tool accepts a string for single recipients and a list of strings for multiple recipients. Separate multiple email addresses with a semicolon. This input can accept a string of the email address or a string of the display name with the email address.

Examples:

bojo@xyz.com

bojo@xyz.com;tsmith@abc.com

Boris Johnson|bojo@xyz.com

Boris Johnson|bojo@xyz.com;tsmith@abc.com

BCC Recipients

Optionally specifies recipients who should receive a "Blind Carbon Copy" of the reply message. This tool accepts a string for single recipients and a list of strings for multiple recipients. Separate multiple email addresses with a semicolon. This input can accept a string of the email address or a string of the display name with the email address.

Examples:

bojo@xyz.com

bojo@xyz.com;tsmith@abc.com

Boris Johnson|bojo@xyz.com

Boris Johnson|bojo@xyz.com;tsmith@abc.com

Reply To

Optionally specifies one or more email addresses to which the recipients of the reply message should direct their own replies. This tool accepts a string for single recipients and a list of strings for multiple recipients. Separate multiple email addresses with a semicolon. This input can accept a string of the email address or a string of the display name with the email address.

Examples:

bojo@xyz.com

bojo@xyz.com;tsmith@abc.com

Boris Johnson|bojo@xyz.com

Boris Johnson|bojo@xyz.com;tsmith@abc.com

Sender

Optionally specifies a local email address from which this reply message should be sent. This input can accept a string of the email address or a string of the display name with the email address.

Examples:

bojo@xyz.com

Boris Johnson|bojo@xyz.com

The email system will generate the reply message in the outbox folder of the mailbox associated with this address, so that it will appear to the recipients as if the message originated from the owner of that mailbox. The transmission of the reply message will

fail if the mailbox associated with this address is not accessible by the email system. If no From address is specified, the message will originate in the outbox folder of the mailbox belonging to the account under which the CIC email system is running. Separate multiple email addresses with a semicolon.

Subject

Specifies the text to appear on the subject line of the email message. You may enter a literal value like "Caller disconnected without leaving a voice mail message" or build a complex expression using the [Expression Editor Assistant](#).

Importance

Specifies the importance of the message as either "Low", "Normal", or "High".

Sensitivity

This parameter is a string that can have the value of "Normal", "Personal", "Private", or "Confidential". The default is "Normal", and this will also be used in the event that something other than one of the other three valid values is entered.

Body

A List of Email Body type. Specifies the text to appear in the body of the email message. You may enter a literal value or build a complex expression using the [Expression Editor Assistant](#). A message might contain multiple bodies, including "text/plain" and "text/html" bodies.

"text/plain" and a "text/html" (if available).

Message Attachments

A list of email cookies containing any attached messages.

Attachment Files

A list of string value containing zero or more complete file paths. You must assign the attachment files (and their paths) as the elements of a list of string variable. You cannot just type the filename and path in this parameter. You can use a Send Email tool step to send a message with a .WAV attachment as a Voice Mail message. Just specify "Voice" as the Content Type of the message. Further, if you check the "Delete Attachments After Sending" checkbox on the Inputs page, CIC will delete the .WAV file from the file system after the message has been successfully transmitted.

Note: OLE attachments (e.g. bitmaps rendered inline) are not visible, as attachments or otherwise, to handlers. As far as the handler tools are concerned, and email message consists of a text body and zero or more *file* attachments. Other sorts of attachments, such as embedded OLE objects and "nested" messages, are not accessible through the handler tools, since they are not supported or easily simulated on messaging systems other than Exchange (i.e. IBM Notes or IMAP).

Delete Attachments

When set to True, all files in the attachments list will be deleted after they have been attached (copied) to the email message and successfully sent. It is sometimes the case, especially when replying to or forwarding a message, that you need to create a temporary file and attach it to the message being sent. After the message has been sent the temporary file needs to be deleted, but since outgoing mail is handled asynchronously in IP, the handler can't delete the file immediately after the tool returns, because the message most likely hasn't yet been processed. This checkbox simply passes the responsibility for deleting the file(s) on to the asynchronous service thread that actually sends the message since that thread is the only one that actually knows when it's safe to delete it (or them).

Delivery Receipt

Set to True to prompt the recipient for a delivery receipt upon receiving this email. If the underlying mail system does not support this feature, it will be ignored.

Read Receipt

Set to True to prompt the recipient for a read receipt upon opening this email. If the underlying mail system does not support this feature, it will be ignored.

Saved Copy

A string designating the mailbox cookie in which copies of the email are saved. This string must be the mailbox moniker as formatted in the Mailbox attribute in DS. For example:

```
ININ.Mail.ObjectMoniker:<x-inin-mail.ex.store:/mailbox/o=Support,ou=Admin%20Group,cn=Recipients,cn=John_Doe>
```

The tool will then save the message in the Sent Items folder (in the case of Microsoft Exchange) of John Doe.

Message Class

The MAPI message class for the message. The message class may be one of "Text", "Voice" or "Fax".

Timeout

The number of seconds the tool will wait for a return value.

Scheduled

The DateTime variable for the future delivery of a message.

Attachment Files

A list of attached file objects output from another tool, such as [Make Attached File](#).

Outputs

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This step takes the Success exit path if the message was successfully queued for sending. This tool does not verify that the message was actually sent or verify that the recipient's address.

Failure

This step takes the Failure exit path if the CIC server is out of memory, indicating a much larger and unrelated problem.

Send Fax

This Email step sends a fax file retrieved with a [Get Fax File](#) step and sends it to one or more recipients as an attachment to an Email. Because this tool sends a message asynchronously, it only queues a message for sending and does not wait to verify that the message was actually sent or if the recipient's address is correct. As a result, this tool takes the Success exit path, even if the message is not sent.

Note: If you want to send a fax that you have converted to a format other than .l3F, including .TIF, use the [Send Email](#) tool.

Inputs

To Recipients

Specifies the recipient of the email. Separate multiple email addresses with a semicolon. Separate email addresses with a semicolon. Separate multiple email addresses with a semicolon. This input can accept a string of the email address or a string of the display name with the email address.

Examples:

bojo@xyz.com

bojo@xyz.com;tsmith@abc.com

Boris Johnson|bojo@xyz.com

Boris Johnson|bojo@xyz.com;tsmith@abc.com

If you leave the To and CC parameters empty, the message is sent to the address specified in the Unaddressed Mail Recipient [server parameter](#).

Note: In Directory Services, each user's Exchange address is stored in that user's User key in the emailAddress attribute. This applies to MS Exchange users only.

CC Recipients

Specifies who should receive a copy of this email. Separate multiple email addresses with a semicolon. This input can accept a string of the email address or a string of the display name with the email address.

Examples:

bojo@xyz.com

bojo@xyz.com;tsmith@abc.com

Boris Johnson|bojo@xyz.com

Boris Johnson|bojo@xyz.com;tsmith@abc.com

BCC Recipients

Optionally specifies recipients who should receive a "Blind Carbon Copy" of the reply message. Separate multiple email addresses with a semicolon. This input can accept a string of the email address or a string of the display name with the email address.

Examples:

bojo@xyz.com

bojo@xyz.com;tsmith@abc.com

Boris Johnson|bojo@xyz.com

Boris Johnson|bojo@xyz.com;tsmith@abc.com

Reply To

Optionally specifies one or more email addresses to which the recipients of the reply message should direct their own replies. Separate multiple email addresses with a semicolon. This input can accept a string of the email address or a string of the display name with the email address.

Examples:

bojo@xyz.com

bojo@xyz.com;tsmith@abc.com

Boris Johnson|bojo@xyz.com

Boris Johnson|bojo@xyz.com;tsmith@abc.com

Sender

Optionally specifies a local email address from which this reply message should be sent. This input can accept a string of the email address or a string of the display name with the email address.

Examples:

bojo@xyz.com

Boris Johnson|bojo@xyz.com

The email system will generate the reply message in the outbox folder of the mailbox associated with this address, so that it will appear to the recipients as if the message originated from the owner of that mailbox. The transmission of the reply message will fail if the mailbox associated with this address is not accessible by the email system. If no From address is specified, the message will originate in the outbox folder of the mailbox belonging to the account under which the CIC email system is running. Separate multiple email addresses with a semicolon.

Subject

Specifies the text to appear on the subject line of the email message. You may enter a literal value like "Caller disconnected without leaving a voice mail message" or build a complex expression using the [Expression Editor Assistant](#).

Importance

Specifies the importance of the message as either "Low", "Normal", or "High".

Sensitivity

This parameter is a string that can have the value of "Normal", "Personal", "Private", or "Confidential". The default is "Normal", and this will also be used in the event that something other than one of the other three valid values is entered.

Body

A List of Email Body type. Specifies the text to appear in the body of the email message. You may enter a literal value or build a complex expression using the [Expression Editor Assistant](#).

Fax ID

The variable that will indicate the name of the Fax File to be attached to the email.

Messages Attachments

A list of email cookies containing any attached messages.

Attachment Files

A list of string value containing zero or more complete file paths. You must assign the attachment files (and their paths) as the elements of a list of string variable. You cannot just type the filename and path in this parameter. You can use a Send Email tool step to send a message with a .WAV attachment as a Voice Mail message. Just specify "Voice" as the Content Type of the message. Further, if you check the "Delete Attachments After Sending" checkbox on the Inputs page, CIC will delete the .WAV file from the file system after the message has been successfully transmitted.

Delete Attachments

When set to True, all files in the Attachments List will be deleted after they have been attached (copied) to the email message and successfully sent. It is sometimes the case, especially when replying to or forwarding a message, that you need to create a temporary file and attach it to the message being sent. After the message has been sent the temporary file needs to be deleted, but since outgoing mail is handled asynchronously in IP, the handler can't delete the file immediately after the tool returns, because the message most likely hasn't yet been processed. This checkbox simply passes the responsibility for deleting the file(s) on to the asynchronous service thread that actually sends the message since that thread is the only one that actually knows when it's safe to delete it (or them).

Delivery Receipt

Set to True to prompt the recipient for a delivery receipt upon receiving this email. If the underlying mail system does not support this feature, it will be ignored.

Read Receipt

Set to True to prompt the recipient for a read receipt upon opening this email. If the underlying mail system does not support this feature, it will be ignored.

Saved Copy

String designating the mailbox cookie in which copies of the email are saved.

Timeout

The number of seconds the tool will wait for a return value.

Scheduled

The DateTime variable for the future delivery of a message.

Attachment Files

A list of attached file objects output from another tool, such as [Make Attached File](#).

Output

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Path

Success

This step takes the Success exit path if the message was successfully queued for sending. This tool does not verify that the message was actually sent or verify that the recipient's address.

Failure

This step takes the Failure exit path if the CIC server is out of memory, indicating a much larger an unrelated problem.

Send Message Light Notification

This Email tool sends a notification to initiate the [Message Light Notification initiator](#). This tool should be used from handlers that add messages to a user's mailbox (such as voice mail or fax delivery) and by handlers that access messages (such as remote voice mail retrieval or remote fax retrieval).

Inputs

User ID

This string value is the CIC User ID of the user who has the message.

Operation

This integer value is 0 if the message was added to the user's inbox, or 1 if the message was read or deleted from the user's inbox.

Unread Count

This integer value is the number of unread email messages in the user's inbox, or -1 if unknown.

Unread Voice

This integer value is the number of unread voice mail messages in the user's inbox, or -1 if unknown.

Unread Fax

This integer value is the number of unread fax mail messages in the user's inbox, or -1 if unknown.

Exit Paths

Success

This step takes the Success exit path if it read the folder.

Failure

This step takes the Failure path if it could not read the folder.

Send Voicemail

This Email tool step converts an CIC Audio Recording to a .WAV file and sends the .WAV file to one or more recipients as an attachment to an Email.

Inputs

To Recipients

Specifies the recipient of the email. This tool accepts a string for single recipients and a list of strings for multiple recipients. Separate multiple email addresses with a semicolon. This input can accept a string of the email address or a string of the display name with the email address.

Examples:

bojo@xyz.com

bojo@xyz.com;tsmith@abc.com

Boris Johnson|bojo@xyz.com

Boris Johnson|bojo@xyz.com;tsmith@abc.com

If you leave the To and CC parameters empty, the message is sent to the address specified in the Unaddressed Mail Recipient [server parameter](#).

Note: In Directory Services, each user's email address is stored in that user's User key in the Mailbox attribute.

CC Recipients

Specifies who should receive a copy of this email. This tool accepts a string for single recipients and a list of strings for multiple recipients. Separate multiple email addresses with a semicolon. This input can accept a string of the email address or a string of the display name with the email address.

Examples:

bojo@xyz.com

bojo@xyz.com;tsmith@abc.com

Boris Johnson|bojo@xyz.com

Boris Johnson|bojo@xyz.com;tsmith@abc.com

BCC Recipients

Optionally specifies recipients who should receive a "Blind Carbon Copy" of the reply message. This tool accepts a string for single recipients and a list of strings for multiple recipients. Separate multiple email addresses with a semicolon. This input can accept a string of the email address or a string of the display name with the email address.

Examples:

bojo@xyz.com

bojo@xyz.com;tsmith@abc.com

Boris Johnson|bojo@xyz.com

Boris Johnson|bojo@xyz.com;tsmith@abc.com

Reply To

Optionally specifies one or more email addresses to which the recipients of the reply message should direct their own replies. This tool accepts a string for single recipients and a list of strings for multiple recipients. Separate multiple email addresses with a semicolon. This input can accept a string of the email address or a string of the display name with the email address.

Examples:

bojo@xyz.com

bojo@xyz.com;tsmith@abc.com

Boris Johnson|bojo@xyz.com

Boris Johnson|bojo@xyz.com;tsmith@abc.com

Sender

Optionally specifies a local email address from which this reply message should be sent. This input can accept a string of the email address or a string of the display name with the email address.

Examples:

bojo@xyz.com

Boris Johnson|bojo@xyz.com

The email system will generate the reply message in the outbox folder of the mailbox associated with this address, so that it will appear to the recipients as if the message originated from the owner of that mailbox. The transmission of the reply message will fail if the mailbox associated with this address is not accessible by the email system. If no From address is specified, the message will originate in the outbox folder of the mailbox belonging to the account under which the CIC email system is running. Separate multiple email addresses with a semicolon.

Subject

Specifies the text to appear on the subject line of the email message. You may enter a literal value like "Caller disconnected without leaving a voice mail message" or build a complex expression using the [Expression Editor Assistant](#).

Importance

Specifies the importance of the message as either "Low", "Normal", or "High".

Sensitivity

This parameter is a string that can have the value of "Normal", "Personal", "Private", or "Confidential". The default is "Normal", and this will also be used in the event that something other than one of the other three valid values is entered.

Body

A List of Email Body type. Specifies the text to appear in the body of the email message. You may enter a literal value or build a complex expression using the [Expression Editor Assistant](#).

Message Attachments

A list of email cookies containing any attached messages.

Attachment Files

A list of string value containing zero or more complete file paths. You must assign the attachment files (and their paths) as the elements of a list of string variable. You cannot just type the filename and path in this parameter. You can use a Send Email tool step to send a message with a .WAV attachment as a Voice Mail message. Just specify "Voice" as the Content Type of the message. Further, if you check the "Delete Attachments After Sending" checkbox on the Inputs page, CIC will delete the .WAV file from the file system after the message has been successfully transmitted.

Delete Attachments

When set to True, all files in the Attachments List will be deleted after they have been attached (copied) to the email message and successfully sent. It is sometimes the case, especially when replying to or forwarding a message, that you need to create a temporary file and attach it to the message being sent. After the message has been sent the temporary file needs to be deleted, but since outgoing mail is handled asynchronously in IP, the handler can't delete the file immediately after the tool returns, because the message most likely hasn't yet been processed. This checkbox simply passes the responsibility for deleting the file(s) on to the asynchronous service thread that actually sends the message since that thread is the only one that actually knows when it's safe to delete it (or them).

Recording ID

The variable that indicates the name of the audio recording to be converted to a .WAV file and attached to the email.

Audio Format

The format to use for audio recording. See [Compress Audio File](#) tool for possible values.

See the *Voice Mail Compression Options* white paper for a more detailed explanation of audio file compression and compression formats.

Normalize

Set this Boolean to True to normalize the audio recording. When a recording is normalized, it is analyzed to determine what the maximum volume level of the audio file is. A value 5% below the maximum value is then used to set the gain value that will bring the maximum volume level up or down to a standard level. This ensures that at a given station, all recordings will play back at the same relative volume. Using a value 5% below the maximum volume to calculate the gain prevents a short burst of static or similar anomalous noise from throwing off this volume adjustment.

Delivery Receipt

Set to True to prompt the recipient for a delivery receipt upon receiving this email. If the underlying mail system does not support this feature, it will be ignored.

Read Receipt

Set to True to prompt the recipient for a read receipt upon opening this email. If the underlying mail system does not support this feature, it will be ignored.

Saved Copy

String designating the mailbox cookie in which copies of the email are saved.

Attachment Files

A list of attached file objects output from another tool, such as [Make Attached File](#).

Timeout

The number of seconds that the tool will wait for a return value.

Scheduled

The DateTime variable for the future delivery of a message.

Outputs

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Path

Success

This step takes the Success exit path if the message was successfully queued for sending. This tool does not verify that the message was actually sent or verify that the recipient's address.

Failure

This step takes the Failure exit path if the CIC server is out of memory, indicating a much larger an unrelated problem.

Send VPIM Message

This Email tool is similar to the [Send Voicemail](#) tool, but sends a VPIM (i.e. RFC 2421) compliant message. Because the message is sent using SMTP regardless of the underlying mail system, a Hostname is required. A Username and Password will be required only if the SMTP server requires authentication.

Inputs

To Recipients

Specifies the recipient of the email. You must specify the full SMTP address (e.g., JohnD@YourOrg.com). If you leave the To and CC parameters empty, the message is sent to the address specified in the Unaddressed Mail Recipient [server parameter](#).

Note: In Directory Services, each user's email address is stored in that user's User key in the mailbox attribute.

CC Recipients

Specifies the recipient of the email. You must specify the full SMTP address (e.g., JohnD@YourOrg.com). If you leave the To and CC parameters empty, the message is sent to the address specified in the Unaddressed Mail Recipient [server parameter](#). Separate multiple email addresses with a semicolon.

BCC Recipients

Optionally specifies recipients who should receive a "Blind Carbon Copy" of the reply message. You must specify the full SMTP address (e.g., JohnD@YourOrg.com). Separate multiple email addresses with a semicolon.

Reply To

Optionally specifies one or more email addresses to which the recipients of the reply message should direct their own replies. This tool accepts a string for single recipients and a list of strings for multiple recipients.

Sender

Specifies a local email address from which this reply message should be sent. The email system will generate the reply message in the outbox folder of the mailbox associated with this address, so that it will appear to the recipients as if the message originated from the owner of that mailbox. The transmission of the reply message will fail if the mailbox associated with this address is not accessible by the email system. If no From address is specified, the message will originate in the outbox folder of the mailbox belonging to the account under which the CIC email system is running. You must specify the full internet address (JohnD@YourOrg.com). Separate multiple email addresses with a semicolon.

Subject

Specifies the text to appear on the subject line of the email message. You may enter a literal value like "Caller disconnected without leaving a voice mail message" or build a complex expression using the [Expression Editor Assistant](#).

Importance

Specifies the importance of the message as "Low," "Normal," or "High."

Sensitivity

This parameter is a string that can have the value of "Normal," "Personal," "Private," or "Confidential." The default is "Normal," and this will also be used in the event that something other than one of the other three valid values is entered.

Recording ID

The file name of the VPIM recording to be sent. This field is not required, but it is recommended that it be used.

Hostname

The SMTP server ID.

Username

Optional field containing the sender's email username and used in the event that the SMTP server requires authentication.

Password

Optional field containing the sender's email password and used in the event that the SMTP server requires authentication.

Timeout

The time in seconds to wait for the VPIM message to be sent. If this period elapses before the message is sent, the tool will take the Failure exit path with a result code indicating a timeout.

Outputs

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This path is taken if the message is successfully sent.

Failure

This path is taken if the operation fails.

Set Out of Office Status

This Email tool sets the Out Of Office (OOO) status for the given Exchange mailbox cookie. This feature can only be used with CIC systems that are connected to an Exchange mail server.

On Notes or Groupwise, this tool will fail and will create an appropriate entry in the Application Event Log.

Note: The handler author must have intimate knowledge of the mail system in use in order to effectively use this tool, as all parameters can change from one application to the next.

Inputs

Mailbox

String representing the mailbox cookie for the mailbox.

Out Of Office

A boolean set to "true" if an OOO status will be set. If you do not want to change the state of the status, do not change this parameter.

Out of Office Text

A string that contains the text of the OOO status. To clear the string value, use an empty string ("") and to use an existing status text, do not change the parameter.

Outputs

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This path is taken if the OOO status is successfully retrieved.

Failure

This path is taken if the operation fails. The most common reason for failure is a lack of OOO support by the mail system.

Update Folder

This Email tool synchronize the contents of a previously opened folder with the actual current contents of the underlying MAPI message store. Use this tool to access messages which have arrived in the open folder since the Open Folder tool was run and to remove deleted messages from the internal message list.

If the folder is updated asynchronously, then no message count will be available.

Inputs

Folder ID

The unique identifier for the folder being updated. This ID is retrieved with an [Incoming Mail](#) initiator, or the [Open Folder](#) tool.

Max Count

The maximum number of messages to open. No value or a value of 0 will retrieve all messages.

Reverse Order

Set this to True to reverse the default order in which the messages are opened by the mail server. The default order is dependant on the underlying mail system.

Reverse Index

Set this to True to reverse the order in which messages are indexed.

Status

Only messages with this status will be searched, can be Read or Unread.

Importance

Only messages with this importance will be searched, can be low, high, or normal.

Sensitivity

This parameter is a string that can have the value of None, Personal, Private, or Confidential. The default is None, and this will also be used in the event that something other than one of the other three valid values is entered.

Message Class

Only messages with this class will be searched, can be text, voice, fax, or ndr.

Subject

Only messages with this subject will be searched. Uses a regular expression.

Sender

Only messages from this sender will be searched (friendly name or email address). Uses a regular expression.

Start Date/Time

Only message received after this date and time will be searched.

End Date/Time

Only messages received before this date and time will be searched.

Asynchronous

Set this to true if you want the tool should to until the messages have been loaded before exiting.

Timeout

The number of seconds the tool will wait for a return value.

Outputs

Unread Count

The total number of unread messages in the folder.

Read Count

The total number of read messages in the folder.

Unread Voice

The total number of unread Voice Mail messages in the folder.

Read Voice

The total number of read Voice Mail messages in the folder.

Unread Fax

The total number of unread Faxes in the folder.

Read Fax

The total number of read Faxes in the folder.

Result

The results of the operation.

See [Email Tool Result Codes](#) for more information.

Exit Paths

Success

This step takes the Success exit path if the folder was updated.

Failure

This step takes the Failure exit path if the mail server is not running.

Fax

Introduction to Fax Tools

The Fax tools are for building handlers that send and receive faxes automatically. (The Interaction Fax viewer is for sending faxes manually.) The documents you fax can be in either [Interaction Fax format](#) (.I3F files) or one of the [Supported Bitmap File Formats](#). If you have a collection of documents that have been converted into a collection of Interaction Fax format documents, you can build a handler that creates faxes containing a specific collection of those documents. The following steps describe of how the fax tools should be used as the basis for building fax-back functionality via handlers.

Typically, the fax tools are used in the following order within a handler, as illustrated by this diagram, entitled [The order in which Fax tools should be used](#). Use this multi-step process as guide when creating fax functionality within a handler.

Step One: Create the Fax Page List

A fax page list defines what will be faxed. The first step in any handler that sends a fax is the [Create Fax Page list](#) tool. This tool specifies the Interaction Fax file or bitmap, and the pages within that file, to send.

Step Two: Append Pages to the Fax Page List (optional)

An optional second step, use one or more [Append Pages](#) steps to add more pages to the fax. Append Pages lets you create a fax from more than one file. This is useful if you want users to be able to request multiple documents in a single phone call.

Step Three: Create the Fax Object

Once you have compiled the fax page list, use a Create Fax tool to convert the list to a fax object that can be processed by [Fax Services](#).

Step Four: Create an Envelope, Print the Fax, or Email the Fax

Once you have a fax object, you either create a fax envelope, print the fax, or email the fax. Creating a fax envelope is simply specifying information about how the fax should be sent, such as the recipient's phone number, information that should go on the cover page, the time to send the fax, and an email address to notify if the fax is sent or if the fax send fails. Fax envelopes are created using the [Create Envelope](#) tool. You can also print the fax to a local or network printer using the [Print Fax](#) tool. Finally, if you want to send the fax as an attachment to an email, you can use the [Send Fax Mail](#) tool. You don't need to create an envelope if you are printing or emailing the fax object.

Step Five: Queue the Fax for Sending

If you want to send the fax via Fax Services, and you have created the fax envelope described in step four, you can queue the fax for sending with the [Queue Fax For Send](#) tool. This tells Fax Services to send the fax using the information specified in the fax envelope. Once a fax has been sent, Fax Services generates an Export Fax File event. This event starts the [Fax Send Completed](#) initiator.

Fax tools:

[Append Document Pages](#)

[Append Pages](#)

[Append Word Pages](#)

[Convert PRN to I3F](#) (The Convert PRN to I3F tool was deprecated in CIC 4.0)

[Create Empty Fax Page List](#)

[Create Envelope](#)

[Create Fax](#)

[Create Fax Page List](#)

[Export Fax File](#)

[Get Fax File](#)

[Print Fax](#)

[Queue Fax For Send](#)

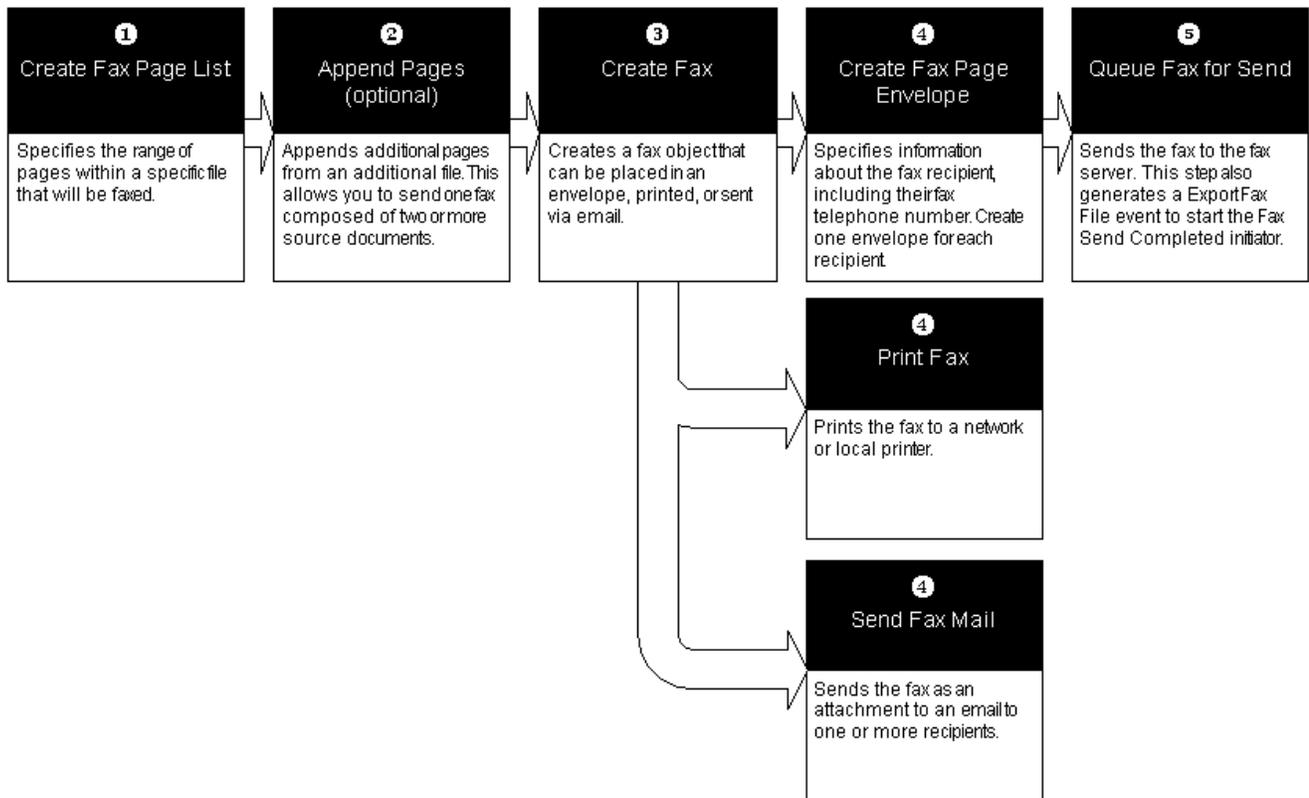
[Receive Fax Call](#)

[Send Fax Now](#)

Fax Tool Usage

This diagram illustrates the order in which the [Fax Tools](#) should be used. For best results when printing, set your paper orientation to Landscape.

The order in which EIC Fax tools should be used



Supported Bitmap File Formats

The following file formats can be sent with the [Create Fax Page List](#) and [Append Page tools](#).

- [JPEG \(JPG\)](#)
- [TIF](#)
- [BMP](#)
- [PCX](#)
- [PCD and FPX](#)
- [WMF](#)
- [PSD](#)
- [PNG](#)
- [TGA](#)
- [EPS](#)
- [RAS](#)
- [WPG](#)
- [PCT](#)
- [MAC, IMG, MSP](#)

Interaction Fax format

Interaction Fax files are documents that can be opened and sent from Interaction Fax or manipulated by the [Create Fax Page List](#) and [Append Page tools](#) in a handler. Interaction Fax files are files created using one of the following methods:

- **Printing a document to the Interaction Fax Print Driver**
When you install a CIC client, the Interaction Fax print driver is added as one of your available printers. You can then print documents from any application to the Interaction Fax print driver. This print driver converts that document to a multipage .I3F file and opens that file in the Interaction Fax application. You can then either specify a recipient, or save the .I3F file to a directory. If you save the .I3F, you can send it from a handler using the [Create Fax Page List](#) and [Append Page tools](#).

Fax Tools

Append Document Pages

This Fax tool provides the ability to use Adobe Acrobat Reader to create faxes. Any file that can be loaded into Adobe Acrobat Reader 5.0 and printed can be converted into a fax using this tool step.

Inputs

Fax Page List

Fax page list created with Create Fax Page List or Create Empty Fax Page List.

Path to page source

String containing any path/filename accepted by Adobe Acrobat Reader. (See Comments below)

Desired resolution

Resolution of the created fax. Can be either 98 for Standard resolution or 196 for Fine resolution.

Desired resolution

Paper size of the created fax. Can be either 0=Letter, 1=Legal, or 2=A4.

Timeout for the Document Renderer to PrintTo

This is the maximum time to wait for Adobe Acrobat Reader to render the fax document in seconds (Default is 3600 seconds or 1 hour). Keep in mind that the size of the document impacts how long it will take for the Adobe Acrobat Reader 5.0 to render the document so make sure to make this value long enough to render the largest Adobe Acrobat Reader document desired.

Exit Paths

Success

This path is taken if the fax is successfully created.

Failure

This path is taken if the operation fails. Check the Event Viewer for details of why this tool step failed.

Comments

This tool step requires the CIC Render Server to be installed.

This tool step can require Adobe Acrobat Reader to be installed on the CIC Render Server depending on CIC Render Server configuration in Interaction Administrator. For more information about CIC Render server configuration in Interaction Administrator, see [Fax Server](#) in Interaction Administrator Help. For more information about the CIC Render server, see the [CIC Render Server Technical Reference](#) in the PureConnect Documentation Library.

The Path to page source input can be anything accepted by the Adobe Acrobat Reader file input dialog. If a relative path is provided then it will be relative to the Resource directory.

The Adobe Acrobat Reader file is not converted into a fax until the Create Fax tool step is executed. Therefore, the Append Document Pages will succeed even with erroneous input values, but the Create Fax tool step will fail. When the Create Fax tool step fails, a Fax event will be logged in the Windows Event Viewer detailing the reason for the failure.

Append Pages

This Fax tool adds pages to a Fax page list that was created with a [Create Fax tool](#). See [Fax tools](#) for more information.

Inputs

Fax Page List

The unique ID for the fax page list to which to add pages.

Location type of fax pages

The type of location where the Interaction Fax format (.I3F) documents reside. You should always specify an integer value of 0. This is because UNC (or File Path) path is the only type currently accepted. In a future release, types such as email message or other formats might be used. Other values are reserved for the future. You may also enter a server or system parameter using the following syntax: \${parameter_name}.

Type of file

Type 0 for an [Interaction Fax format](#) (.I3F files), 1 for one of the [Supported Bitmap File Formats](#), 2 for a printable document file.

Note: The 32-bit version of the application associated with the document type (Word, Excel, PowerPoint, or Acrobat) must be installed on the CIC Server. Documents created in a version previous to Office 97 will not work because they do not support the ActiveX Printable Document interface. Documents must be converted to Office 97 before they can be used with this tool.

First Page to Retrieve

Type 1 to start at the first page, or specify a page within file.

Last page to retrieve

The last page of the document to retrieve. Enter a page number greater than First Page to Retrieve, or enter -1 to use all pages from First Page to Retrieve to the end of the document.

Path to page source

Specify a path and file name. If you specify only a file name, the default path is value in the Work Directory [server parameter](#) (set in Interaction Administrator). If you want to use a path other than the default path, use a fully qualified UNC path. You may also enter a server or system parameter using the following syntax: \${parameter_name}.

Caution: If you specify a file not on the local server, you may experience a noticeable delay if that server or named file is not available.

Exit Paths

Success

The Success path is take if the page list is created successfully.

Failure

The Failure exit path is taken if there was not enough free memory to create the page list or if any input parameters are invalid.

Append Word Pages

This Fax tool provides the ability to use Microsoft Word to create faxes. Any file that can be loaded into Microsoft Word and printed

can be converted into a fax using this tool step. In addition, custom document property settings can be passed into the document and a user defined macro can be run before creating the fax.

Note: Ensure that in the copy of Microsoft Word on the server, the "Update Fields" setting is on. This is found in Word in the **Tools>Options** dialog on the Print tab.

Inputs

Fax Page List

Fax page list created with Create Fax Page List or Create Empty Fax Page List.

Path to page source

String containing any path/filename accepted by Microsoft Word. (See Comments below)

First page to retrieve

First page from the Word document that will be converted to this fax.

Last page to retrieve

Last page from the Word document, or -1 for all remaining pages.

Desired resolution

Resolution of the created fax. Can be either 98 for Standard resolution or 196 for Fine resolution. See comments.

Run Word Macro

Runs the named Word macro after setting the custom document properties, but before converting to fax. See the Microsoft Word Basic "Run" command for proper syntax.

List of Property Names to pass to Word

String list of Word custom document properties that will be set. The List of Property Values must contain the desired values, in the same order.

List of Property Values to pass to Word

String list of values corresponding to the List of Property Names to be set in the Word custom document properties.

Exit Paths

Success

This path is taken if the fax is successfully created.

Failure

This path is taken if the operation fails. Check the Event Viewer for details of why this tool step failed.

Comments

This tool step requires Microsoft Word 97 or later to be installed on the CIC server. Changes to the Word file (via the custom document properties or the macro) are reflected in the resulting fax but not saved in the Word file.

The Path to page source input can be anything accepted by the Word file input dialog. This can vary by the version of Word. For example, Microsoft Word 2000 can accept URLs to valid documents and web pages while Microsoft Word 97 cannot.

Microsoft Word does not properly implement the ActiveX Printable Document standard at this time, preventing CIC from controlling the resolution used in the creation of the fax. Although this tool step accepts a resolution, it is ignored by Word, which instead uses the resolution set in the Document Defaults for the Interaction Fax print driver. This can be changed by opening the Printer applet, right-mouse clicking the Interaction Fax print driver and selecting the Document Defaults... menu item.

The Word file is not converted into a fax until the Create Fax tool step is executed. Therefore, the Append Word Pages will succeed even with erroneous input values, but the Create Fax tool step will fail. When the Create Fax tool step fails, a Fax event will be logged in the Windows Event Viewer detailing the reason for the failure (for example, an invalid macro name).

Convert PRN to I3F

The Convert PRN to I3F tool was deprecated in CIC 4.0.

Create Empty Fax Page List

Use this Fax tool when you need to send only a cover page. With this tool, you can create an empty page list, use that to create a fax. After that, Envelopes are created with the [Queue Fax for Send](#) tool step, where you can specify cover page information. See [Fax tools](#) for more information.

Outputs

Fax Page List

The identifier for the empty fax page list.

Exit Paths

Success

The Success exit path is taken if the empty fax page list is created.

Failure

The Failure exit path is taken only if the CIC Server fails.

CreateEnvelope

This Fax tool creates a fax envelope for the fax. A fax envelope must be created for each fax recipient. See [Fax tools](#) for more information.

Inputs

Fax ID

The fax to be sent to the address in this envelope.

Note: All other inputs should be ignored because they will be overwritten by the inputs to the Queue Fax for Send tool step.

Recipient Fax Number

The fax recipient's phone number. A comma causes a two-second pause. Any numbers that follow the "/" symbol are dialed after the call is connected (used to dial an extension, for example).

Recipient Company Name

The fax recipient's company name.

Recipient Name

The fax recipient's name.

Sender Name

The sender's name.

Sender's Fax Number

The sender's fax number.

Sender's Telephone Number

The sender's telephone number.

Sender's Company Name

The sender's company name.

Notify sender when fax is sent

Set this to true if you want the sender to be notified if the fax was sent successfully.

Email address to notify if fax when fax is sent

The sender's email address.

Notify sender if fax fails

Set this to true if you want to sender to be notified if the fax was not sent successfully.

Email address to notify if fax cannot be sent

The sender's email address, or some other address to notify if the fax was not sent successfully.

Cover Page Comments

Comments to add to the cover page.

Page Header Message

Header information for the cover page.

Cover page Type

Only Interaction Fax cover page documents (with a file extension of .i3c) are valid in this release. This may change in future releases.

Cover page Name

The base name (no path or extension) of a file in the CoverPages subdirectory under the IC resources directory. So, if you have a cover page file called I3Confidential.i3c, you would place it on the CoverPages directory under the IC resources directory and type the name "I3Confidential" in this parameter.

Fax Workstation Group Name

The fax station on which to send the fax.

Number of Retries to Attempt

The number of times CIC will attempt to send the fax.

Delay between retries

How long to wait (in seconds) between retries.

Maximum BPS to attempt

The maximum BPS to attempt. Usually you should set this to 0.

Send Type

The category of time when the fax should be sent.

Scheduled Time

The time the fax should be sent if the Send type is one.

Time of day low rates begin

The begin time a fax should be sent if the Send Type was two.

Time of day low rates end

The latest time a fax should be sent if the Send Type was two.

Login name user or handler that created this fax

The network user id of the person or handler sending the fax.

Outputs

Fax Envelope ID

The identifier of the newly created envelope. This can be used with Queue Fax For Send to initiate a fax send.

Exit Paths

Success

The Success exit path is taken if the envelope was created successfully.

Failure

The Failure exit path is taken if the envelope was not created successfully. This can occur when a Fax ID is not valid.

Create Fax

This Fax tool creates a fax from a list of page sources created using the Create Fax Page List, Create Empty Page List and Append Pages tools. An empty page list can be used to send a fax consisting only of a cover page. See [Fax tools](#) for more information.

Inputs

Fax Page List

The page list created Create Fax Page List, Create Empty Page List and Append Pages tools. The pages used and their order are defined by the order these tools were used to create the page list.

Outputs

Fax Id

The identifier for the fax that was created as a result of this tool.

Exit Paths

Success

The Success exit path is taken if the fax is created successfully.

Failure

The Failure exit path is taken if any parameters used to create the page list are invalid. This includes wrong or invalid Type of file, invalid page ranges and invalid paths to the page source files.

Create Fax Page List

This Fax tool creates a list of document pages to include in a fax. Once the fax page list is created with this tool, you may later append additional pages using the [Append Pages](#) tool. See [Fax tools](#) for more information.

Inputs

Location type of fax pages

The type of location where the Interaction Fax format (.I3F) documents reside. You should always specify an integer value of 0. This is because UNC (or File Path) path is the only type currently accepted. In a future release, types such as email message or other formats might be used. Other values are reserved for the future. You may also enter a server or system parameter using the following syntax: \${parameter_name}.

Type of file

Type 0 for an [Interaction Fax format](#) (.I3F files), 1 for one of the [Supported Bitmap File Formats](#), or 2 a printable document file.

Note: The 32-bit version of the application associated with the document type (Word, Excel, PowerPoint, or Acrobat) must be installed on the CIC Server. Documents created in version previous to Office 97 will not work because they do not support the ActiveX Printable Document interface. Documents must be converted to Office 97 versions before they can be used with this tool.

First page to retrieve

Type 1 to start at the first page, or specify a page within file.

Last page to retrieve

The last page of the document to retrieve. Enter a page number greater than First Page to Retrieve, or enter -1 to use all pages from First Page to Retrieve to the end of the document.

Path to page source

Specify a path and file name. If you specify only a file name, the default path is value in the Work Directory [server parameter](#) (set in Interaction Administrator). If you want to use a path other than the default path, use a fully qualified UNC path. You may also enter a server or system parameter using the following syntax: \${parameter_name}.

Caution: If you specify a file not on the local server, you may experience a noticeable delay if that server or named file is not available.

Outputs

Fax Page List

The identifier for the fax page list this tool creates.

Exit Paths

Success

The Success exit path is taken if the page list is created successfully.

Failure

The Failure exit path is taken if there was not enough free memory to create the page list or if any input parameters are invalid.

Export Fax File

This Fax tool exports Interaction Fax files to one of several common image file formats. Supported formats include:

- TIFF (using either Group 3 or Group 4 fax compression)
- DCX (the multi-page version of PCX)

IMPORTANT - Once the file is exported, it is the responsibility of the handler writer to delete the file. Exported files are not automatically cleaned up when the handler finishes. Also, care should be taken when emailing the file to always use the email tool's feature to delete the file after sending, instead of deleting the file yourself.

Inputs

Fax ID

ID of the fax to be exported.

Export File Type

Integer representing the desired export file type as follows:

- 0 - TIFF, Group 3 compression
- 1 - TIFF, Group 4 compression
- 2 - DCX

Path and filename to create

Specify a path and file name. If you specify only a file name, the default path is the value in the Work Directory [server parameter](#) (set in Interaction Administrator). If you want to use a path other than the default path, use a fully qualified UNC path. You may also enter a server or system parameter using the following syntax: \${parameter_name}.

Caution: If you specify a file not on the local server, you may experience a noticeable delay if that server or named file is not available.

Outputs

Export File Pathname

String containing the path and filename the fax was exported to.

Exit Paths

Success

This step takes the Success exit path if the fax is successfully exported.

Failure

This step takes the Failure exit path if the export file cannot be created due to an invalid FaxId or invalid export file name.

Get Fax File

This Fax tool retrieves a fax file so that it can be sent via email. Creates a copy of the fax file to a location so that it can be attached to an email sent by the [Send Fax Mail](#) tool. When a handler containing the Get Fax File tool ends, the copy of the file is removed from the output directory. See [Fax tools](#) for more information.

Inputs

Fax Id

The identifier for the fax to be converted to a file.

Export File Type

Integer representing the type of file that the fax will be converted into.

Integer Corresponding File Type

- 0 Tiff Group 3
- 1 Tiff Group 4
- 2 DCX
- 3 PNG
- 4 I3F (default)
- 5 PDF

Outputs

Fax File Id

The identifier for the fax file that this tool created. The fax file created is a temporary copy and is deleted when the handler exits.

Fax File Path and Name

This information is used by the Send Fax Mail tool to find and attach the file. The fax file created is a temporary copy and is deleted when the handler exits.

Exit Paths

Success

The Success exit path is taken if the fax was retrieved.

Failure

The Failure exit path is taken if the fax file was not created successfully. This can occur when a Fax Id is not valid.

Print Fax

This Fax tool outputs a fax to a network or local printer. See [Fax tools](#) for more information.

Inputs

Fax Id

The identifier of the fax to be printed.

Printer Name

The name of the printer. For locally attached printers, this is the name of the printer listed in the Printers folder. For remote printers, this consists of the server name and the name listed in the Printers folder. For example, if you have a printer installed as HPLaserJet4M and the printer is actually attached to the CompanyPrintServer computer, then the full printer name is \\CompanyPrintServer\HPLaserJet4M.

Pagination

Shrink will resize each fax page to fit the printed page. If the fax page is larger than the printed page, any text or images in the fax will be reduced. Chop will remove any portions of each fax page that do not fit on the printed page. If the Spill option is selected, each fax page will result in as many printed pages as necessary to contain the original fax image at its original size.

Exit Paths

Success

The Success exit path is taken if the fax was queued for printing successfully.

Failure

The failure path is taken if there is an error queuing the print job. This can occur when a Fax ID is not valid. If an invalid printer name is supplied, this tool step will take the Success path, but an error will be logged in the NT Event Viewer.

Queue Fax For Send

This Fax tool sends a Fax to the CIC fax server. Most of the information used for queuing is the same information used to create the fax envelope. See [Fax tools](#) for more information.

Inputs

Fax Envelope Id

The Envelope Id created for the fax to be queued by a [Create Fax Envelope](#) step.

Recipient Fax Number

The fax recipient's phone number. A comma causes a two-second pause, and any numbers after the "/" symbol are dialed after the call is connected.

Recipient Company Name

The fax recipient's company name.

Recipient Name

The fax recipient's name.

Sender Name

The sender's name.

Sender's Fax Number

The sender's fax machine telephone number.

Sender's Telephone Number

The sender's telephone number.

Sender's Company Name

The sender's company name.

Notify Sender when fax is sent

Set this to true if you want the sender to be notified if the fax was sent successfully.

Email address to notify if fax when fax is sent

The sender's email address.

Notify Sender if fax fails

Set this to true if you want the sender to be notified if the fax was not sent successfully.

Email address to notify if fax cannot be sent

The sender's email address, or some other address to notify if the fax was not sent successfully.

Cover Page Comments

Comments to add to the cover page.

Page Header Message

Header information for the cover page.

Cover Page Type

Only Interaction Fax cover page documents (with a file extension of .i3c) are valid in IC release 1.0. This may change in future releases.

Cover page Name

This is the base name (no path or extension) of an CIC cover page (.i3c) file. The file must reside in the \\3\IC\Server\Resources\Coverpages folder on the CIC server.

Fax Workstation Group Name

The fax workstation on which to send the fax.

Number of Retries to Attempt

How many times CIC will attempt to send the fax.

Delay between retries

How many seconds to wait between retries.

Maximum BPS to attempt

The maximum BPS to attempt. Usually you should set this to 0.

Send Type

The category of time when the fax should be sent.

0 = As soon as possible

1 = At a scheduled time

2 = During a low rate time period

Scheduled Time

The time the fax should be sent if the Send type is one.

Time of day low rates begin

The begin time a fax should be sent if the Send Type is two.

Time of day low rates end

The latest time a fax should be sent if the Send Type is two.

Login name user or handler that created this fax

The network user id of the person sending the fax.

Station Id

The name of the sending fax station to use.

Call Attribute Names

This takes a list of strings (array) of attribute names, but you can use a single name. This might be used for tracking Account Code attributes on faxes.

Call Attribute Values

This takes a list of strings (array) of attribute values, but you can use a single value to match a single attribute name. This might be used for tracking Account Code attribute values on faxes.

Line groups

The line group to use to place a fax call.

List of dial strings to be used

The list of dial strings to use for a fax call.

Intercom call?

Flag to indicate that the fax call should be placed as an intercom (internal) call as opposed to an outbound call.

Queue to dial for Intercom calls

If the Intercom call flag indicates that the fax call should be placed as an intercom call, this value represent the internal queue to use.

Exit Paths

Success

The Success exit path is taken if the fax was queued successfully.

Failure

The Failure exit path is taken if the fax was not queued successfully. This can occur if the fax envelope no longer exists or if the recipient information does not exist.

Receive Fax Call

This Fax tool instructs the CIC fax server to attempt to receive a fax on this call. This tool can be used with the [Extended Get Key](#) tool as the next step after the Extended get key step detects a tone. See [Fax tools](#) for more information.

Inputs

Call Identifier

The identifier for the fax call.

Fax Workstation Group Name

The name of the Fax Station Group that should be used to receive this fax. 'All' means that any available Fax Group can be used.

Exit Paths

Success

The Success exit path is taken if the server has an available device in the specified Fax Workstation Group to receive the fax.

Failure

The Failure exit path is taken if the fax server is unable to receive the fax at this time. This could can occur when all the devices in the specified Fax Workstation Group are in use, or the call is terminated before a device became available.

Send Fax Now

This Fax tool allows a handler to transmit a fax on an existing call. This makes it possible to implement a fax-back system using the call from the original caller. To send a fax on an existing call, it is necessary to have an active call (as a Call ID) and a fax object (as a Fax ID). The Send Fax Now tool step will not return until the fax is transmitted or there was an error. When the Send Fax Now tool step returns, the call is still active and it is the handler's responsibility to disconnect the call when finished with it. See [Fax tools](#) for more information.

Inputs

Fax Id

The identifier for the fax that was created by the Create Fax tool.

Call Identifier

Call Id of an active call. The caller should have an attached fax machine set to receive a fax.

Fax Workstation Group Name

Name of the fax device group to use to send this fax. Using the group "*" will use any available fax device.

Timeout in seconds

Time in seconds to wait for an available device in the specified fax device group.

Maximum BPS

Maximum BPS to negotiate with remote fax machine. Valid values are 14400, 12000, 9600, 4800, 2400, 1200, 600, 300 or 0 to negotiate the highest speed supported by both sides.

Station Id

Sets the station name displayed on the fax page and on the remote fax machine's display. This should usually be set to your fax phone number.

Page Header Message

Message to be displayed at the top of each page.

Login Name User or Handler That Created This Fax

The CIC username to be associated with this fax.

Cover Page Type

This input is set for future use. For now, this must be 0.

Cover Page Name

Name of the cover page template to use. Leave this blank if no cover page is desired. Cover page names consist of just the base filename part of the cover page file. For example, to use the cover page template FaxCoverPage.i3c, use "FaxCoverPage". All cover page files must reside in the Cover Pages directory under the IC Resources directory.

Optional Inputs

Recipient Company Name

Recipient Name

Recipient Voice Number

Recipient Fax Number

Sender's Company Name

Sender Name

Sender's Telephone Number

Sender's Fax Number

Cover Page Comments

Outputs

Station Id

Station Id of the remote fax machine.

Fax Device

Name of the fax station used to transmit the fax.

Call Duration

Length of the fax transmission in seconds.

Pages Transmitted

Number of pages (including cover page) sent to remote fax machine.

Fax Transmission Rate

Baud rate used to transmit the fax.

Exit Paths

Success

The Success exit path is taken if the fax is successfully transmitted to the recipient.

Invalid Group

The fax group name specified in the "Fax Workstation Group Name" input is not a valid group. Check the fax groups in IA for a valid group or use the "*" group.

Timeout

The number of seconds specified in the "Timeout" input have elapsed without obtaining an available fax device.

Fax Failed

Fax transmission was not completed successfully. Some pages may have been transmitted to remote. This error is often due to line noise or the remote user interrupting the fax transmission. Check the event log for errors.

Fatal Error

The fax failed for some other reason. Check the event log for errors.

Feedback

Introduction to Feedback Tools

The Feedback tools are used with the Interaction Feedback module. Click on one of the tools below for more information about that tool.

[Get Active Survey](#)

[Get Survey TUI XML](#)

[IsSurveyActive](#)

[Manage Survey Prompt Result](#)

[Post Survey Results](#)

[Run Survey](#)

[Transfer to Survey Queue](#)

[Wait for Survey Start](#)

Get Active Survey

This Feedback tool uses the Interaction ID and workgroup to determine from the Survey engine whether or not to play a customer satisfaction survey after the agent disconnects.

Inputs

Interaction ID

The interaction ID to analyze.

Workgroup Name

The name of the workgroup queue associated with the interaction.

Outputs

Survey ID

The ID of the returned survey. This field is empty if there is no active survey.

Last modified time

The date/time stamp for the survey. This data is used to update the TUI XML cache.

Record Call

A value indicating whether the call should be recorded for this survey.

Exit Code

Exit code associated with any error condition.

Exit Text

Text of the error message.

Exit Paths

Success

This path is taken if the determination is made successfully.

Failure

This path is taken if the operation fails.

Get Survey TUI XML

This Feedback tool gets the TUI XML that will be used to present the customer satisfaction survey.

Inputs

Interaction ID

The ID of the interaction for which the tool is to retrieve the TUI XML.

Survey ID

The ID of the survey to present.

Outputs**Survey TUI XML**

The XML to play for the survey.

Invitation Prompt File

The path for the invitation prompt .wav file.

Invitation Text-to-Speech

Text defined for the invitation prompt.

Opt-In Prompt File

Path for the Opt-in prompt .wav file.

Opt-In Text-to-Speech

Text defined for the Opt-in prompt.

Opt-Out Prompt File

Path for the Opt-out .wav file.

Opt-Out Text-to-Speech

Text defined for the Opt-out prompt.

Exit Code

The exit code associated with any error condition.

Exit Text

The text of the error message.

Exit Paths**Success**

This path is taken if the TUI XML is retrieved successfully.

No License

This path is taken if there is no license available for an inbound IVR survey.

Failure

This path is taken if the operation fails.

IsSurveyActive

This Feedback tool clears the TUI XML cache after a period of time to free up deactivated customer satisfaction surveys.

Inputs

List of SurveyIds

The list of surveys to check.

Outputs

List of Booleans

A list indicating which of the checked surveys are active.

Exit Code

The exit code associated with any error condition.

Exit Text

The text of the error message.

Exit Paths

Success

This path is taken if the TUI XML cache is successfully cleared.

Failure

This path is taken if the operation fails.

Manage Survey Prompt Result

This Feedback tool allows the handlers to pass a result back to the Survey engine to indicate that a prompt was recorded.

Inputs

Prompt Recording SessionId

The name of the file that contains the prompt recording.

Outputs

Exit Code

The exit code associated with any error condition.

Exit Text

The text of the error message.

Exit Paths

Success

This path is taken if the results are successfully passed back to the Survey engine.

Failure

This path is taken if the operation fails.

Post Survey Results

This Feedback tool sends the customer satisfaction survey results to the Survey engine to go into the database.

Inputs

Interaction ID

The interaction on which the survey was executed.

Survey ID

The unique ID assigned to the survey.

Question Names

Question IDs parallel to the answers.

Question Values

Parallel list of answers.

Event Code

Possible values: InsufficientResources, Complete, OptOut, Error, Abandoned

Outputs

Exit Code

The error code associated with any error condition.

Exit Text

The text of the error.

Exit Paths

Success

This path is taken if the survey results are successfully passed to the Survey engine.

Failure

This path is taken if the operation fails.

Run Survey

This CS Survey tool determines whether the Survey ID is set on the interaction, and if so, triggers the Run Survey initiator.

Inputs

Interaction ID

The call ID assigned to the interaction.

Outputs

Exit Code

Exit code associated with any error condition.

Exit Text

Text of the error message.

Exit Paths

Success

This path is taken if the determination is made successfully.

Failure

This path is taken if the operation fails.

Transfer Survey Queue

This Feedback tool provides the ability to transfer an interaction into the survey queue so a customer satisfaction survey can be run on the interaction.

Inputs

Interaction ID

The call ID assigned to the interaction.

Outputs

Exit Code

Exit code associated with any error condition.

Exit Text

Text of the error message.

Exit Paths

Success

This path is taken if the interaction is successfully transferred into the survey queue.

Not Answered

This exit path is taken if the call was not answered by an agent.

Failure

This path is taken if the operation fails.

Wait for Survey Start

This Feedback tool is used to stop processing a handler until a survey is initiated on the call (Event: eCSSurveyEvent_InitiateSurvey).

Inputs

Interaction ID

The interaction for which a survey will be initiated.

Timeout

The timeout, in seconds, to wait for a survey to start (-1 indicates no timeout).

Outputs

Exit Code

Exit code associated with any error condition.

Exit Text

Text of the error message.

Exit Paths

Success

This path is taken if the survey initiate has fired and there were no errors.

Timeout

This path is taken if the timeout period was reached.

Failure

This path is taken if there was an error waiting for a survey start.

File I/O

Introduction to File I/O Tools

Some File I/O tools create, read, and write to text files. For example, some non-CIC software can utilize or output text files. The text files created, read, or written to could be shared by CIC and the non-CIC software. Other File I/O tools provide TCP/IP functionality. See TCP/IP tools overview for a information on how TCP/IP tools operate.

Click one of the following tools for more information:

[Copy File](#)

[Directory List](#)

[File Attributes Modify](#)

[File Attributes Query](#)

[FileClose](#)

[FileDelete](#)

[File Find Replace](#)

[FileOpenRead](#)

[FileOpenWrite](#)

[FilePosition](#)

[FileRead](#)

[FileReadList](#)

[FileWrite](#)

[FileWriteList](#)

[Get File Statistics](#)

[Get Free Space](#)

[TCP Close](#)

[TCP Connect](#)

[TCP Listen](#)

[TCP Read Integer](#)

[TCP Read String](#)

[TCP Read String UTF8](#)

[TCP Write Integer](#)

[TCP Write String](#)

[TCP Write String UTF8](#)

TCP/IP tools overview

The TCP/IP tools allow handlers to connect and communicate with other's applications on a LAN, WAN, or over the Internet. As long as the remote machine has a valid IP address and the remote application is ready, the TCP/IP tools can send and receive strings and integers.

The TCP/IP tools are typically used in the following order within a handler. As shown in this diagram entitled [The order in which TCP/IP tools should be used](#). Use this multi-step process as guide when creating TCP/IP functionality within a handler.

Step One: Open a Connection

Whether you are connecting to a remote computer or the remote computer is connecting to CIC, you must first establish a connection. When you establish the connection, the resulting connection handle is used by the read and write tools to communicate with the remote computer. If you are connecting to a remote computer, use the [Tcp Connect](#) tool. If the remote computer is connecting to CIC, use the [Tcp Listen](#) tool to monitor a port. When the remote computer makes a connection, the Tcp Listen tool generates an event that starts the [Tcp Connection Accepted](#) initiator. This initiator generates the connection handle.

Step Two: Send and Receive Strings and Integers

Once the connection handle is established, the [Tcp Read String](#) and [Tcp Read Integer](#) tools can take data sent by the remote computer and place it in a variable to be used within the handler. [Tcp Write String](#) and [Tcp Write Integer](#) can send data to the remote computer. To read and write UTF8-encoded strings, use [Tcp Read String UTF8](#) and [Tcp Write String UTF8](#).

Step Three: Closing the Connection

When the handler has finished sending and receiving data, we recommend that you close the connection with the remote computer. The [Tcp Close](#) tool closes the connection stored in the connection handle.

Copy File

This File I/O tool moves or copies a file from one directory to another.

Inputs

Source Filename

The path and name of the file to be copied or moved.

Destination Filename

The path and name of the file to be created by the operation.

Delete source file on success

Check this option to remove the old file after the file has been copied.

Overwrite destination file if it exists

Check this option to overwrite any files with the same name as the Destination Filename.

Exit Paths

Success

This step takes the Success exit path if the file is copied or moved.

Failure

This step takes the Failure exit path if the path or filename is invalid.

Directory List

This File I/O tool, given an absolute, relative, or UNC path, returns a list of files contained in a specified directory.

Inputs

Directory Path

The path to, and name of, a directory. This value can be an absolute path, a relative path, or a UNC path.

Directory Mask

Use this parameter to pass in any wildcards. Acceptable wildcards include the * symbol and the ? symbol. * represents one or more of any character, ? represents a single character. If you wanted a list of all handlers in a specified directory, you would type *.ihd in the Directory Mask field. If you wanted all handlers that started with the letter A, then you would type A*.ihd in the Directory Mask field. If you wanted all handlers with five character file names that start with A, you would type A????.ihd in the Directory Mask field.

By default, the Directory Mask value is *.* , returning all files in the specified directory.

Outputs

Directory List

A list of string value containing the names of any files in the specified directory. These files match any Directory Mask criteria.

Exit Paths

Success

This step takes the Success exit path if either of the following is true:

- The directory is found and is not empty.
- The directory exists, but is empty, and a Directory Mask input value of *.* is specified.

Failure

This step takes the Failure path if either of the following is true:

- The directory does not exist.
- The directory exists, but is empty, and a Directory Mask input value of *.pdf, *.txt, etc., is specified.

File Attributes Modify

This File I/O tool changes the values of several file attributes. You can retrieve a file's attributes with the [File Attributes Query](#) tool.

Inputs

Source Filename

The name of the file to query. Specify a fully qualified path.

Normal

True means the file has no other attributes set. False if the file has other attributes set.

Read-Only

True means the file is read-only, false means the file is not read-only.

Hidden

True means the file is hidden, false means the file is not hidden.

System

True means the file is a system file, false means the file is not a system file.

Volume

The name associated with the drive on which this file is located.

Directory

The directory on which the file is located.

Archive

True means the file is archived; false means the file is not archived.

Exit Paths

Success

This tool takes the Success exit path if the file was found and the attribute values were retrieved.

Failure

This tool takes the Failure exit path if the file could not be found.

File Attributes Query

This File I/O tool returns several file attributes. You can change a file's attributes with the [File Attributes Modify](#) tool.

Inputs

Source Filename

The name of the file to query. Specify a fully qualified path.

Outputs

Normal

True means the file has no other attributes set. False if the file has other attributes set.

Read-Only

True means the file is read-only, false means the file is not read-only.

Hidden

True means the file is hidden, false means the file is not hidden.

System

True means the file is a system file, false means the file is not a system file.

Volume

The name associated with the drive on which this file is located.

Directory

The directory on which the file is located.

Archive

True means the file is archived; false means the file is not archived.

Exit Paths

Success

This tool takes the Success exit path if the file was found and the attribute values were retrieved.

Failure

This tool takes the Failure exit path if the file could not be found.

File Close

This File I/O tool closes a file previously opened with a [FileOpenRead](#) or [FileOpenWrite](#) step.

Inputs

File Handle

The handle for the file you want to close. This handle was created in a FileOpenRead or a FileOpenWrite step.

Exit Paths

Success

This step takes the Success exit path if the operation is successful.

Failure

This step takes the Failure exit path if the file handle is invalid.

File Delete

This File I/O step deletes a file. The deleted file should not be open for read or write when it is deleted. If the file does not exist, this step fails.

Inputs

File Name

The name of the file to be deleted.

Exit Paths

Success

This step takes the Success exit path if the operation is successful.

Failure

This step takes the Failure exit path if the file handle is invalid.

File Find Replace

This File I/O tool finds the first or all occurrences of a specific string in a file and replaces them with a specific string. This tool accepts any document type.

Inputs

Source file

The fully qualified path to and name of the file to process.

Find

The string to find. You may not specify the "/n" character in the string to search for.

Replace

The string used to replace the string specified in Find.

Replace All

Set this value to True to replace all occurrences. Set this value to false to replace only this first occurrence.

Result file

When processing is complete, the new file is saved with the file name you specify here. You must specify a fully qualified path.

Outputs

Replaced occurrences

The number of replacements that were made.

Error messages

The reason this tool failed. Below is a list of possible explanations:

Error Message	Explanation
'Find' string cannot contain an \n character.	The string you specified in Find contained a newline ("/n") character.
"Success"	This tool executed successfully.
FileIO error(%d) on file(%s)	An error occurred while processing the file.
Fatal Error	Some other error has occurred. Make sure the file exists.

Exit Paths

Success

This tool takes the Success exit path if the operation completes successfully.

Failure

This tool takes the Failure exit path if the process failed. See the Error Message output parameter for an explanation of returned error codes.

File Open Read

This File I/O tool opens a file for read access. Tools that can access this file opened for reading are [FileRead](#), [FileReadList](#), [FilePosition](#), and [FileClose](#).

Inputs

File Name

The name of the file to be opened for read access.

Outputs

File Handle

The handle created for the file that was opened for read access.

Exit Paths

Success

This step takes the Success exit path if the operation is successful.

Failure

This step takes the Failure exit path if the file name is invalid or if the file is already open.

File Open Write

This File I/O tool opens a file for write access. Tools that can access this file opened for writing are [FileWrite](#), [FileWriteList](#), [FilePosition](#), and [FileClose](#).

Inputs

File Name

The name of the file to be opened for write access.

Append to existing file?

Check this option to append the new text to the end of the file. Setting this parameter to false means that new text overwrites any existing text starting at the beginning of the file.

Outputs

File Handle

The handle created for the file that was opened for write access.

Exit Paths

Success

This step takes the Success exit path if the operation is successful.

Failure

This step takes the Failure exit path if the file name is invalid or if the file is already open.

File Position

This File I/O tool controls the current byte offset position in a file.

Inputs

File Handle

The handle for a file created during a [FileOpenRead](#) or [FileOpenWrite](#) step.

File Position

A byte offset in the file to which the position is to be set. This is an integer variable.

File Position Type

Determines how file position is interpreted. If set to zero, file position is the byte offset from the beginning of the file. If set to one, file position is the byte offset in relation to the current position. If set to two, the file position is the byte offset relative to the end of the file.

Outputs

New File Position

An integer variable whose value is set to the current byte offset position in the file when the operation is complete.

Note: File size can be determined by setting the file position to zero and file position type to one. Then examine the value of New File Position.

Exit Path

Next

This step always takes the Next exit path.

File Read

This File I/O tool reads a string from a file.

Inputs

File Handle

The handle for a file created during a [FileOpenRead](#) step.

String Delimiters

The code or codes that indicate the end of a string, **any one of which will be interpreted as the end of the string**. The default is "\n" which is a common code for line-break. Possible values are:

" "	space
""	empty string
"\""	quotation mark
"\n"	line break
"\t"	tab

Outputs

String Read From File

A string variable whose value is set to the data read from the file.

Delimiter which ended read operation

This variable contains the value of the delimiter that ended the read operation.

Exit Paths

Success

This step takes the Success exit path if the operation is successful.

Failure

This step takes the Failure exit path if the file handle is invalid.

File Read List

This File I/O tool reads more than one string from a file and writes those strings to a variable of type string list.

Inputs

File Handle

The handle for a file created during a [FileOpenRead](#) step.

String delimiters

The code or codes that indicates the end of a string. The default is "\t" which is a common code for a tab stop.

Possible values are:

" "	space
""	empty string
"\"	quotation mark
"\n"	line break
"\t"	tab

Delimiter to end ReadList operation

The code or codes that indicates the end of the text to be read. This step stops reading text when this code is reached. Other possible values are the same as those listed for the previous parameter.

Outputs

Strings Read From File

A List of String variable whose value is set to the data read.

Delimiter which ended read operation

This variable contains the value of the delimiter that ended the read operation. The default is "\n" which is a common code for line-break.

Possible values are:

" "	space
""	empty string
"\"	quotation mark
"\n"	line break
"\t"	tab

Exit Paths

Success

This step takes the Success exit path if the operation is successful.

Failure

This step takes the Failure exit path if the file handle is invalid.

File Write

This File I/O tool writes a string to a file.

Inputs

File Handle

The handle for a file created during a [FileOpenWrite](#) step.

String to write to file

The string to be written to the file.

Delimiter to append to string

The code or codes to append to the end of the text being written. The default is "\n" which is a common code for line-break. Possible values are:

" "	space
""	empty string
"\"	quotation mark
"\n"	line break
"\t"	tab

Exit Paths

Success

This step takes the Success exit path if the operation is successful.

Failure

This step takes the Failure exit path if the file handle is invalid.

File Write List

This File I/O tool writes the contents of a List of String variable to a text file.

Inputs

File Handle

The handle for a file created during a [FileOpenWrite](#) step.

Strings to write to file

The strings to be written to the file.

String delimiters

The code or codes to append to the beginning of the text being written. The default is "\t" which is a common code for tab stop.

Possible values are:

" "	space
""	empty string
"\"	quotation mark
"\n"	line break
"\t"	tab

Delimiter to end operation

This variable contains the delimiter to be appended to the end of the text written. The default is "\n" which is a common code for line-break. The possible values are the same as those listed for the previous parameter.

Exit Paths

Success

This step takes the Success exit path if the operation is successful.

Failure

This step takes the Failure exit path if the file handle is invalid.

Get File Statistics

This File I/O tool retrieves various file properties for a specified file or directory.

Inputs

Source Filename

The path and file name to return properties for the file, or only the path to return properties for the directory. This value can be an absolute path, a relative path, or a UNC path.

Outputs

Full Path

The complete absolute path, and file name, for the specified file.

File Size

The file's size in bytes.

Creation Time

The date and time the file was created. This is a DateTime value.

Modify Time

The date and time the file was last modified. This is a DateTime value.

Access Time

The date and time the file was last accessed. This is a DateTime value.

Exit Paths

Success

This step takes the Success exit path if the file or directory is found.

Failure

This step takes the Failure path if the file does not exist.

Get Free Space

This File I/O tool returns the amount of unused space on a specific drive.

Inputs

Root Directory of drive to check

The root directory of the drive to check. Be sure to include the \ symbol after the drive letter. For example, to check the size of C:, you would type: C:\

Outputs

Free Space in KB

The size in KB of the available space on the specified drive. 1 KB = 1024 bytes.

Exit Paths

Success

This step takes the Success exit path if the drive is found.

Failure

This step takes the Failure path if the drive does not exist.

TCP Close

This File I/O tool closes a specified TCP/IP connection with another computer. See [TCP/IP tools](#) for more information on using TCP/IP tools.

Inputs

Socket Handle

The identifier for the socket connection to be closed.

Exit Paths

Next

This tool always takes the Next exit path.

TCP Connect

This File I/O tool opens a TCP/IP connection with another computer. You can connect to a computer on the local network (Marketing), an IP address (172.16.120.10), or a full host name (ftp.inter-intelli.com). See [TCP/IP tools](#) for more information on using TCP/IP tools.

Inputs

Server ID

The host name or IP address of the computer to which you want to connect.

TCP Port for connection

The port number on the remote machine to which you want to connect.

Outputs

Socket handle

The unique identifier for this established connection. Other TCP/IP tools use this handle for sending and receiving strings and integers.

Exit Paths

Success

This tool takes the Success exit path if the connection is established and a handle is returned.

Failure

This tool takes the Failure exit path if a connection cannot be established.

TCP Listen

This File I/O tool tells CIC to start listening for another computer attempting to make a TCP/IP connection with the CIC server. When a connection is established, this tool generates a TCP/IP Connection event, which can be used to start the [TCP/IP Connection Accepted](#) initiator. See [TCP/IP tools](#) for more information on using TCP/IP tools.

Inputs

TCP Port for connection

The port to watch for a connection.

Exit Paths

Success

This tool takes the success path if the port is successfully identified and watched.

Failure

This tool takes the Failure exit path if another TCP Listen tool has already been executed to watch the specified port.

TCP Read Integer

This File I/O tool reads an integer value passed in by another computer over a TCP/IP connection.

Inputs

Socket handle

A unique socket connection identifier generated by the [TCP Connect](#) or [TCP/IP Connection Accepted](#).

Integer byte ordering

Different machine architectures sometimes store data using different byte orders. For example, Intel-based machines store data in the reverse order of Macintosh (Motorola) machines. If you are not sure of the byte ordering of the machine sending you the integer value, leave this value at the default true.

Read timeout

The number of seconds to wait for the integer before taking the Timeout exit path.

Outputs

Integer value

The read integer value.

Exit Paths

Success

This tool takes the Success exit path if the connection is established and a value is returned.

Timeout

This tool takes the Timeout exit path if nothing was read within the number of seconds specified in the Read timeout parameter.

Failure

This tool takes the Failure exit path if the socket handle is invalid.

TCP Read String

This File I/O tool reads a string value passed in by another computer over a TCP/IP connection.

Inputs

Socket handle

A unique socket connection identifier generated by the [TCP Connect](#) or [TCP/IP Connection Accepted](#).

String type

The type of string being read.

Number of characters to read

The maximum number of characters to read before taking the Success exit path. If the end of string character is read before reaching the maximum, this tool stops reading and takes the Success exit path.

Characters to indicate end of string

The character that denotes the end of the string. If the maximum number of characters is read before reading the end of string character, this tool stops reading and takes the Success exit path. If multiple characters are assigned, this tool will recognize only the first assigned character. For example, if "end" is configured to indicate the end of the string, this tool will recognize only the "e" as the end of string character and will stop reading the first time it encounters an "e."

Other characters you might want to use include are listed below. CIC interprets the following characters when this tool executes. For example, if you specify "\t," a tab indicates the end of the string.

"\n"	CIC interprets this as a new line
"\r"	CIC interprets this as a carriage return
"\t"	CIC interprets this as a tab character
" "	CIC interprets this as a space
"\""	CIC interprets this as a quotation mark

Read timeout

The number of seconds to wait for the integer before taking the Timeout exit path.

Outputs

String value

The read string value.

Exit Paths

Success

This tool takes the Success exit path if the connection is established and a value is returned.

Timeout

This tool takes the Timeout exit path if nothing was read within the number of seconds specified in the Read timeout parameter.

Failure

This tool takes the Failure exit path if the socket handle is invalid.

TCP Read String UTF8

This File I/O tool reads a UTF8-encoded string value passed in by another computer over a TCP/IP connection.

Inputs

Socket handle

A unique socket connection identifier generated by the [TCP Connect](#) or [TCP/IP Connection Accepted](#).

Number of characters to read

The maximum number of characters to read before taking the Success exit path. If the end of string character is read before reaching the maximum, this tool stops reading and takes the Success exit path.

Characters to indicate end of string

The character that denotes the end of the string. If the maximum number of characters is read before reading the end of string character, this tool stops reading and takes the Success exit path. If multiple characters are assigned, this tool will recognize only the first assigned character. For example, if "end" is configured to indicate the end of the string, this tool will recognize only the "e" as the end of string character and will stop reading the first time it encounters an "e."

Other characters you might want to use include are listed below. CIC interprets the following characters when this tool executes. For example, if you specify "\t," a tab indicates the end of the string.

"\n"	CIC interprets this as a new line
"\r"	CIC interprets this as a carriage return
"\t"	CIC interprets this as a tab character
" "	CIC interprets this as a space
"\""	CIC interprets this as a quotation mark

Read timeout

The number of seconds to wait for the integer before taking the Timeout exit path.

Outputs

String value

The read string value.

Exit Paths

Success

This tool takes the Success exit path if the connection is established and a value is returned.

Timeout

This tool takes the Timeout exit path if nothing was read within the number of seconds specified in the Read timeout parameter.

Failure

This tool takes the Failure exit path if the socket handle is invalid.

TCP Write Integer

This File I/O tool writes an integer value to the computer connected via TCP/IP. See [TCP/IP tools](#) for more information on using TCP/IP tools.

Inputs

Socket handle

A unique socket connection identifier generated by the [TCP Connect](#) or [TCP/IP Connection Accepted](#).

Integer value

The value you want to write to the connection.

Integer byte ordering

Different machine architectures sometimes store data using different byte orders. For example, Intel-based machines store data in the reverse order of Macintosh (Motorola) machines. If you are not sure of the byte ordering of the machine receiving the integer value, leave this value at the default true.

Exit Paths

Success

This tool takes the Success exit path if the integer was successfully written.

Failure

This tool takes the Failure exit path if the socket handle is invalid.

TCP Write String

This File I/O tool writes a string value to the computer connected via TCP/IP. See [TCP/IP tools](#) for more information on using TCP/IP tools.

Inputs

Socket handle

A unique socket connection identifier generated by the [TCP Connect](#) or [TCP/IP Connection Accepted](#).

String value

The value you want to write to the connection.

String type

The type of string being written.

Exit Paths

Success

This tool takes the Success exit path if the string was successfully written.

Failure

This tool takes the Failure exit path if the socket handle is invalid.

TCP Write String UTF8

This File I/O tool writes a UTF8-encoded string value to the computer connected via TCP/IP. See [TCP/IP tools](#) for more information on using TCP/IP tools.

Inputs

Socket handle

A unique socket connection identifier generated by the [TCP Connect](#) or [TCP/IP Connection Accepted](#).

String value

The value you want to write to the connection.

Exit Paths

Success

This tool takes the Success exit path if the string was successfully written.

Failure

This tool takes the Failure exit path if the socket handle is invalid.

Generic Object

Introduction to Generic Objects Tools

The Generic Objects tools are for creating, transferring, disconnecting, and otherwise manipulating objects.

Click on a tool below to learn more about that tool:

[Create Generic Object](#)

[Disconnect Generic Object](#)

[Get Generic Object Attributes](#)

[Set Generic Object Attributes](#)

[Transfer Generic Object](#)

Create Generic Object

This Generic Object tool creates an object ID in the specified queue.

Inputs

Queue Name

The workgroup or user queue which contains the object. The fully qualified queue name must be used. For example, "Workgroup Queue:Marketing" or "User Queue:John Doe."

Direction of Interaction

Enter "1" to indicate an incoming interaction, and "2" for an outgoing interaction.

Remote Party Type

Enter "1" to indicate an internal party, and "2" for an external party.

Local Party Type

Enter "1" to indicate an internal party, and "2" for an external party.

Remote Name

The value entered here is used to populate the **Name** field on the My Interactions view in the CIC client.

Remote ID

The value entered here is used to populate the **Number** field on the My Interactions view in the CIC client.

This populates the "line" field on the various lines pages on the Interaction Client

The value entered here is used to populate the **Line** field on various pages in the CIC client.

Outputs

Generic Object ID

The identifier for the specified object.

Exit Paths

Success

This step takes the Success exit path if the object was successfully created.

Failure

This step takes the Failure exit path if the object could not be created. This may occur if the queue name is incorrect or if the object does not exist.

Disconnect Generic Object

This Generic Object tool removes a generic object.

Inputs

Generic Object ID

The identifier for the specified object.

Disconnection Type

Enter 0 if the generic object is being disconnected by an internal party, an enter 1 if the generic object is being disconnected by an external party.

Exit Paths

Success

This step takes the Success exit path if the object was successfully disconnected.

Failure

This step takes the Failure exit path if the object could not be disconnected. This may occur if the ID of the object is incorrect, the object has already been disconnected or it does not exist.

Get Generic Object Attributes

This Generic Object tool retrieves one attribute from a specified object.

Inputs

Generic Object ID

The identifier for the specified object.

Attribute Name

The name of the attribute this step retrieves.

Outputs

Attribute Value

The variable that contains the value of the retrieved attribute.

Exit Paths

Success

This step takes the Success exit path if the specified attribute value was successfully retrieved.

Failure

This step takes the Failure exit path if it was unable to retrieve the attribute value. This may occur if the ID of the object is incorrect or if it does not exist.

Set Generic Object Attributes

This Generic Object tool sets the value of an attribute for a specified object.

Inputs

Generic Object ID

The identifier for the specified object.

Attribute Name

The name of the attribute to set.

Attribute Value

The variable that contains the value of the attribute to set.

Exit Paths

Success

This step takes the Success exit path if the specified attribute value was successfully set.

Failure

This step takes the Failure exit path if it was unable to set the attribute value. This may occur if the ID of the object is incorrect or if it does not exist.

Transfer Generic Object

This Generic Object tool moves a generic object to the specified work group or user queue.

Inputs

Generic Object ID

The identifier for the specified object.

Queue Name

The queue to which the generic object will be transferred.

Exit Paths

Success

This step takes the Success exit path if the object was successfully transferred to the specified queue.

Failure

This step takes the Failure exit path if the object did not transfer to the specified queue. This may have occurred because either the object ID was incorrect, or the specified queue does not exist.

Host Interface

Introduction to Host Interface tools

Note: Interaction Host Recorder does not natively include SSL encryption and authentication support. In CIC 3.0, you could use NetManage (now MicroFocus) OnWeb Web-to-Host to take advantage of these SSL options. MicroFocus no longer packages the required DLLs for SSL encryption and authentication support in their product line. Therefore, Interaction Host Recorder no longer supports these SSL options. CIC and PureConnect releases after CIC 4.0 do not include the Host Server license and do not support the ability to use the SSL encryption and authentication. Genesys recommends that you convert to a SOAP or REST solution instead of using Host Interface tools.

CIC has several tools that allow CIC to communicate with mainframes via the TN 3270(E) and TN 5250 protocols. The Host Server, an CIC Server subsystem, can log on to and perform operations through a terminal emulation. A handler containing the Host tools can tell the Host Server to start a Telnet session with a mainframe (or AS400) and pass information to and from via screen scrapes. This is similar to performing a database put or get, but you are performing the operation through a mainframe terminal emulation running on the Host server.

Screen scraping is the process by which information is read from or to the fields in a terminal emulation. This process does not introduce any security risks because the Host Server logs in just like any other terminal emulation user does, allowing mainframe administrators to use existing security precautions.

Licensing for Host Interface Tools

Starting in IC 2.3, you must purchase an CIC license for the desired number of "Host access tool sessions," which come in bundles of 10, 24, or 50, depending on how many simultaneous connections to a mainframe or other computer you need to support. Once you have the license file containing a Host Server license key, load it on the CIC server in Interaction Administrator using the File... License Management menus. Once loaded, you will see the I3_LICENSE_HOST_SERVER(x) entry in the License Management dialog, where x is the number of licenses you purchased.

Configuring Host Server

Part of the configuration for the Host Server Interface is in the System Configuration container in Interaction Administrator. The Host Server tab lets you configure the maximum number of concurrent sessions and the path to the host tools. See the Host Server Configuration online help in Interaction Administrator for details on these options.

How does it work?

The CIC Host tools instruct the Host Server to create and maintain a terminal connection with the mainframe, input data to the terminal emulation, and to process output scraped from the terminal emulations. Screen scrapers have three advantages:

- They don't require modification to the mainframe
- You can use the existing business and data integrity logic on the host
- They can be used when no other (peer-to-peer) interface method is available

The diagrams later in this document illustrate the relationship between host server and mainframe.

When would I use these Host tools?

You should use these tools if you have data stored on a mainframe that you want to pass into a handler via terminal emulation, or if you want to pass data from a handler to a mainframe via terminal emulation. Both scenarios might be useful if you want to maintain terminal emulation security protocols or if you do not want to use an ODBC driver to access the data.

One situation in which you might want to use the Host tools is an IVR. You may have account information stored on a mainframe. Using the Host tools, you could look at that information from within a running handler and read it to a person on the telephone, or format the text for an email or fax-back. In another example, you could write values from a handler in the fields of a data entry screen. This might be useful with a web form that allowed users to update information stored on a database. The uses for the tools are limited only by your creativity.

What are the components that make up this functionality?

Three components provide the Host tool functionality. First are the tools themselves that send and receive data, and tell the Host

server what to do. Second is the Host Server, the module that actually creates and maintains connections with the mainframe or AS400. Third is the Interaction Host Recorder, an application you use to create the host profiles (scripts) that drive the terminal emulation. These pieces are described below.

Host tools

The Host tools allow handlers to connect and interact with the Host Server, writing and retrieving information from the terminal emulation. Several tools are designed to navigate between screens and search for specific strings, just like an end user would. Strings retrieved from screens can be used in handlers, and strings created in handlers can be written to screens.

Host Server

This CIC Server subsystem creates and maintains one or more terminal emulation connections with a mainframe or AS400. At this time, the number of connections is limited by licensing restrictions to either 10, 25, or 50 connections. The Host server can optionally cache a connection to improve performance.

When commanded to do so by a Host tool running a handler, the Host server can retrieve values from the fields in the emulated terminal. This is called a screen scrape. You determine which fields are scraped by creating a host profile in the Interaction Host Recorder application.

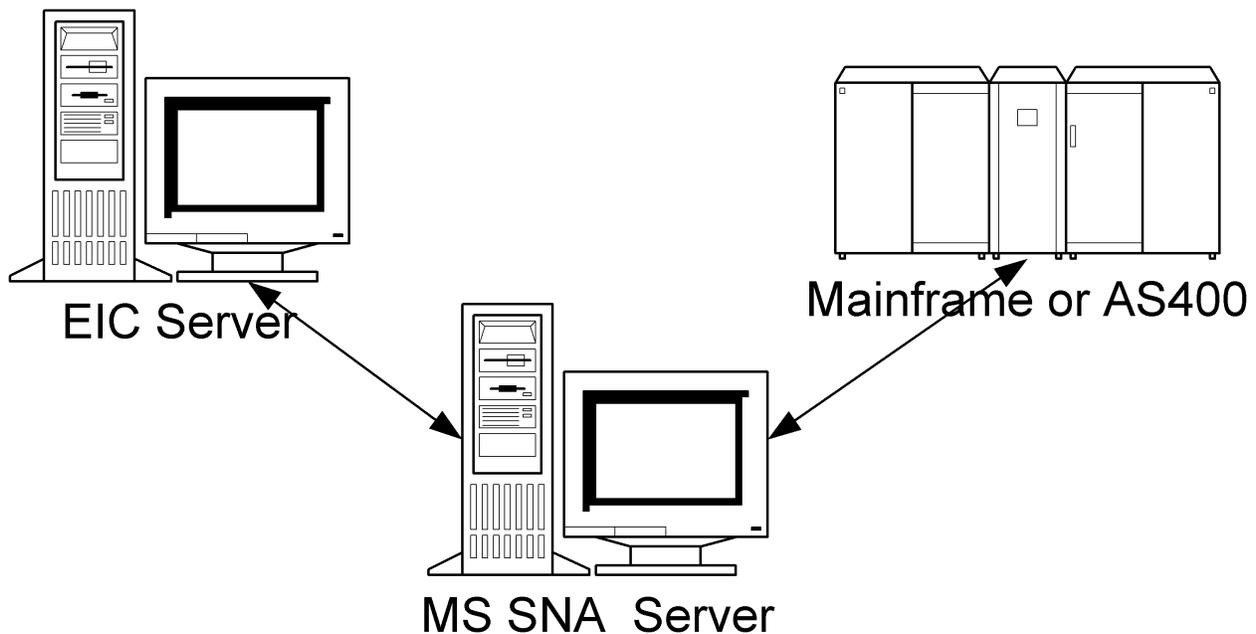
Interaction Host Recorder

Use the Interaction Host Recorder to create the host profiles that log on and navigate between screen in the terminal emulation. The Host Server is able to maintain its orientation once a connection is established, so Host tools can run various parts of the host profile to navigate between screens. See the Host Recorder documentation for more information on creating scripts.

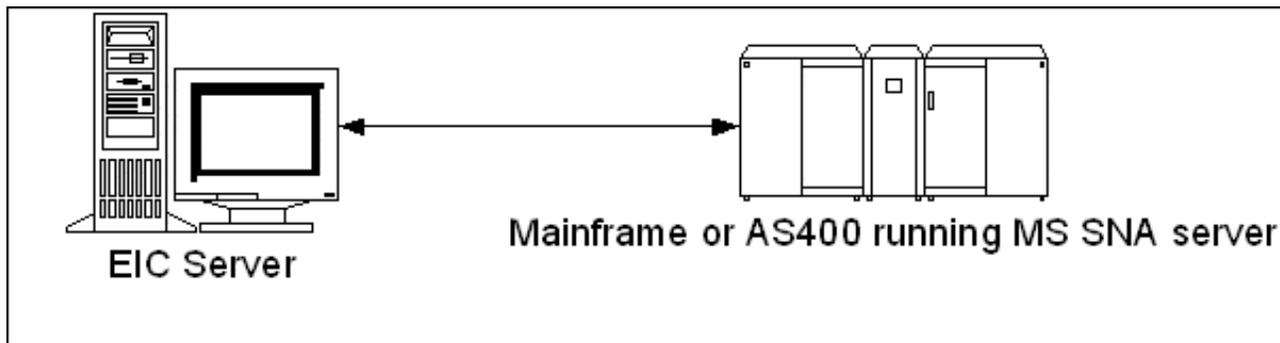
For a complete description of how to use the Interaction Host Recorder to create host profiles, see the online help that accompanies the Interaction Host Recorder.

How should I set up my CIC server to use these tools?

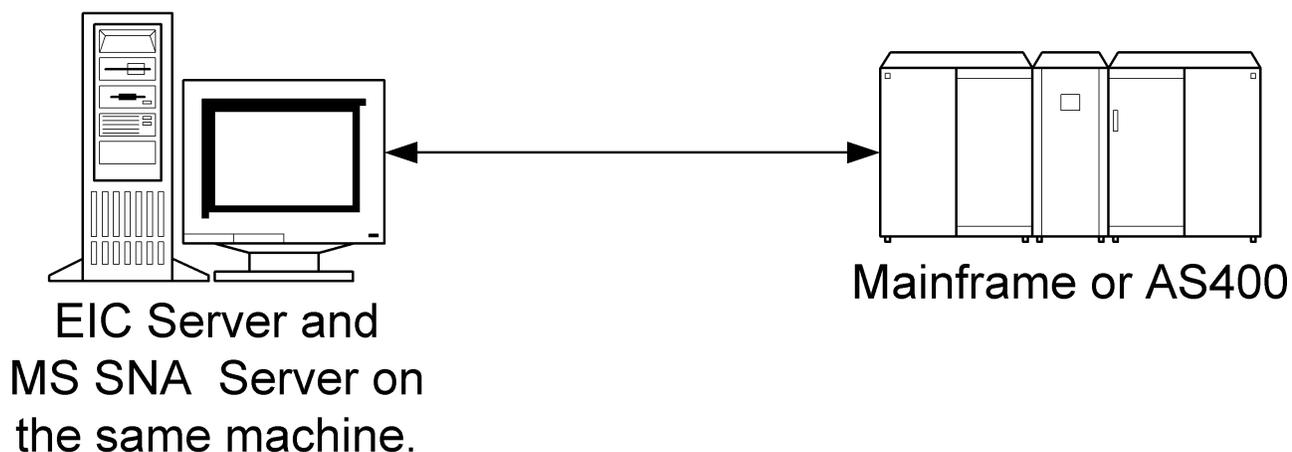
There are two ways to set up your CIC server to communicate with a mainframe via the Host server.



The first method is the recommended method because it does not involve running additional applications on the CIC server. This method is diagramed below:



Another possible configuration which may improve performance over the first implementation is to run the SNA and CIC server on the same machine. In this configuration, the CIC server can access the SNA exposed API to retrieve data directly from the mainframe. The problem with this configuration is that you have to run non-CIC processes on the CIC server, reducing its reliability. This implementation is shown in the following figure:



Host Interface Tools

Click on one of the following Host Interface tools for more information on that tool:

[Host Connect](#)

[Host Connect Ex](#)

[Host Disconnect](#)

[Host Disconnect Ex](#)

[Host Fetch Form Data](#)

[Host Find String](#)

[Host Get Connection Resource](#)

[Host Get Cursor Position](#)

[Host Get Field Attributes](#)

[Host Get Resource Counts](#)

[Host Get Screen](#)

[Host Get Screen Dimensions](#)

[Host Get String](#)

[Host Initialize Resource](#)

[Host Move To Position](#)

[Host Move To Screen](#)

[Host Press Key](#)

[Host Put Form Data](#)

[Host Put String](#)

[Host Verify Screen](#)

[Host Wait For Cursor](#)

[Host Wait For ReadyToSend](#)

[Host Wait For Screen](#)

[Host Wait For String](#)

Host Connect

This Host Interface tool step establishes a connection with the host mainframe. (Actually it tells the [Host Server](#) to establish a connection with the host (mainframe or AS400). It returns a host connection handle that can be used by other Host Tools in this handler or a subroutine running off this handler. Host Connect is typically the first Host Interface tool used when starting a session.

If a Host Disconnect step from a previously running handler cached its connection, this tool can use that cached connection to quickly establish communication with the host. Cached connections are effective when handlers are running so frequently that the host doesn't time out the connection.

For more information on using Host tools, see [Overview of Host Interface tools](#).

Inputs

Host Profile

The name of the Host Profile to use when establishing this connection. See [host profile](#) for more information. This profile contains the connection settings.

Use cached connection if available

Once an open connection is cached, other Host Connect steps in other instances of this handler can quickly connect to the host over this same connection. This enables other handlers to run faster because they do not have to log in to the host. You should choose this option if the handler is going to run frequently and the host won't have time to time out the connection.

Outputs

Host Connection

The handle for the connection to the host mainframe. Other host tools can use this handle when sending commands to the [Host Server](#).

Exit Paths

Success

This step takes the Success exit path if a connection with the host is established and no cached connection was available.

Cache

This step takes the Cache exit path if a cached connection was established.

Failure

This step takes the Failure exit path if a normal or cached connection was not established. This occurs if the mainframe disconnects for some reason.

Host Connect Ex

This Host Interface tool creates a connection to a host and returns a connection handle. If enabled to do so, this tool will also return a resource along with the connection handle.

Inputs

Host Profile

The name of the Host Profile to use when establishing this connection. This profile contains the connection settings.

Use cached connection if available

Once an open connection is cached, other Host Connect steps in other instances of this handler can quickly access the host over this same connection. This enables other handlers to run faster because they do not have to connect to the host.

Screen Name

The optional name of a screen defined in the host profile. This is ignored if "Use cached connection if available" is unchecked. Only those cached connections with matching screen names *and* matching profile names will be eligible to be selected from the cache. If the screen name is empty, then any cached connection with a matching profile will be selected. If a (non-empty) matching screen name is found, then a VerifyScreen is performed internally. If the VerifyScreen fails, the connection will be physically disconnected and purged from the cache and the search for an eligible connection will repeat until one is found or until there are no more connections to check, in which case a new connection is physically created.

Retry Count

The number of times the tool will attempt to create the connection before taking the Failure exit path.

Retry Delay (seconds)

The number of seconds the tool will wait between each attempt to create a connection.

Resource pool name

If this option is selected, the tool will return a resource along with the connection handle once the connection is established.

Outputs

Host Connection

The handle for the connection to the host mainframe. Other host tools can use this handle when sending commands to the Host Server.

Resource

If "Associate a resource with this connection" is selected, this field will contain the resource associated with the host connection.

Exit Paths

Success

This step takes the Success exit path if a connection with the host is established and no cached connection was available.

Cache

This step takes the Cache exit path if a cached connection was established.

Failure

This step takes the Failure exit path if a normal or cached connection was not established.

Host Disconnect

This Host Disconnect tool drops a connection established with a Host Connect step. Typically, this is the last step in a series of Host Interface steps. If you choose to cache the connection, the connection is maintained for other Host Connect steps to connect with the mainframe.

If you plan to use cached connections, consider using a [Host Move to Screen](#) step to navigate back to a starting point before disconnecting from the host. This ensures that handlers using cached connections always start at the same screen.

For more information on using Host tools, see [Overview of Host Interface tools](#).

Inputs

Host Connection

The name of the host connection to drop. The [Host Connect](#) step creates this handle.

Place in connection cache

Makes this connection available to Host Connect steps connecting via cached connections.

Exit Paths

Next

This step always takes the Next exit path.

Host Disconnect Ex

This Host Interface tool ends a connected session with the host with a connection resource.

Inputs

Host Connection

The name of the Host Connection profile to use when establishing this connection. This profile contains the connection settings.

Place in connection cache

Makes this connection available to Host Connect steps connecting via cached connections. If this is not checked, then the connection will be physically disconnected, and any resource associated with the connection will be released.

Screen Name

The name of the screen that the connection is currently on, as defined in the host profile.

Exit Paths

Next

This tool always takes the Next exit path.

Host Fetch Form Data

This Host Fetch Form Data tool retrieves data from a screen. The fields from which this tool collects data are determined by a [host profile](#) which you create. This tool is typically used after navigating to a screen using a [Host Move to Screen](#) step.

Settings Page

Host Connection

The name of the connection created by the [Host Connect](#) tool.

Profile

The name of the host profile you created that defines the screens and fields from which this tool will retrieve data.

Screen Name

The name of the screen defined in the host profile.

Field Bindings

This list contains the field names defined for this screen in the host profile. You can bind variables from your handler to these fields, allowing you to process info from the screen within a handler.

Field

A list of fields for the specified screen defined in your host profile.

Variable

A list of variables bound to these fields. When this step executes, these variables will contain the values retrieved from the fields.

Binding button

Use this button to bind a variable to a field or edit an existing binding.

Clear All Bindings button

Use this button to unbind all variables from all fields.

Exit Paths

Success

This tool takes the Success exit path if the data was retrieved.

Failure

This tool takes the Failure exit path if the Host Connection, Profile, or Screen name are invalid. This tool also fails if the host profile is out of sync with the field definitions used in the handler.

Host Find String

This Host Interface tool returns the row and column coordinates for an instance of a string you specify. Searching begins at row 1, column 1, unless you select the Reverse search option where it begins at the lower right corner.

Inputs

Host Connection

The name of the connection created by the [Host Connect](#) tool.

String to Find

The string to search for on the screen.

Number of occurrences

The number of occurrences to search for before returning the string position. For example, if you specify 2, this tool returns the coordinates for the beginning of the second match.

Case-sensitive Search

Select this option if you want to search for an exact match to the specified string. Do not select this option if you are not concerned with an exact match on case.

Reverse Search

Select this option if you want to start searching at the lower-right coordinate and search backwards. You might use this option if there are multiple occurrences of an item on the screen, and you want to find the last one. Or, perhaps you know that an item will be in the lower right quadrant of the screen, so a reverse search will be (slightly) faster than a forward search.

Outputs

Row

The row coordinate for the match's first character.

Column

The column coordinate for the match's first character.

Exit Paths

Success

This step takes the Success exit path if a match was found.

Failure

This step takes the Failure exit path if the host connection is invalid, or becomes invalid during the search. This occurs if the mainframe disconnects for some reason.

Not Found

This step takes the Not Found exit path if a match was not found between the search string and a string in the screen searched.

Host Get Connection Resource

This Host Interface tool returns a resource associated with a given host connection.

Inputs

Host Connection

The name of the connection created by the [Host Connect Ex](#) tool. Note that the Host Connect Ex tool must have the "Associate a resource with this connection" box checked in order for there to be a resource associated with the host connection.

Outputs

Resource

The resource associated with the host connection.

Exit Paths:

Success

This path is taken if the resource was successfully retrieved.

Failure

This path is taken if an associated resource cannot be retrieved.

Host Get Cursor Position

This Host Interface tool returns the current row and column coordinate of a cursor in a screen.

Inputs

Host Connection

The name of the connection created by the [Host Connect](#) tool.

Outputs

Row

The row coordinate of the cursor.

Column

The column coordinate of the cursor.

Exit Paths

Success

This step takes the Success exit path if the cursor was found.

Failure

This step takes the Failure exit path if the host connection is invalid, or becomes invalid during the search. This occurs if the mainframe disconnects for some reason.

Host Get Field Attributes

This Host Interface tool returns various field attributes for a given host connection, screen, and field.

Inputs

Host Connection

The name of the connection created by the [Host Connect](#) or the [Host Connect Ex](#) tool.

Screen

The name of a screen as defined in the host profile.

Field Name

The name of the field in the screen that is being queried by this tool.

Outputs

Start X

The position on the screen denoting the beginning X coordinate of the field.

Start Y

The position on the screen denoting the beginning Y coordinate of the field.

End X

The position on the screen denoting the ending X coordinate of the field.

End Y

The position on the screen denoting the ending Y coordinate of the field.

Length

The length of the field as calculated using the Start X, Start Y and End X, End Y parameters. Some fields may extend beyond a given line so both coordinates (x,y) are included when determining the length. Each coordinate may contain one character, so a field of n length could contain n characters.

Foreground Color

String denoting the background color of the field.

Background Color

String denoting the background color of the field.

Protected

A Boolean variable indicating whether or not the field is protected.

Bold

A Boolean variable indicating whether or not the text in the field is bolded.

Reverse Video

A Boolean variable indicating whether or not the field is displayed as reverse video.

Underlined

A Boolean variable indicating whether or not the field is underlined.

Blinking

A Boolean variable indicating whether or not the field is blinking.

Hidden

A Boolean variable indicating whether or not the field is hidden.

Exit Paths**Success**

This path is taken if the field attributes are successfully retrieved.

Failure

This path is taken if the operation fails.

Host Get Resource Counts

This Host Interface tool queries the resource pool for the number of available and total resources.

Inputs

Resource Pool Name

The name of the resource pool to query.

Outputs

Available Number

The number of available resources in the resource pool

Total Number

The total number of resources in the resource pool.

Exit Paths

Success

This path is taken if the available and total resource counts are successfully retrieved.

Failure

This path is taken if the tool was unable to retrieve the resource counts.

Host Get Screen

This Host Interface tool retrieves an entire screen and assigns each line as an element in a list of strings. Line 1 would be in element 0, line 2 in element 1, and so on. You can then use the list tools or an expression to pull values from the list. Any blank lines result in an empty string ("").

Inputs

Host Connection

The name of the connection created by the [Host Connect](#) tool.

Outputs

Result List

The list of string containing the list.

Exit Paths

Success

This step takes the Success exit path if the lines of the screen are successfully written to the list of string.

Failure

This step takes the Failure exit path if the host connection is invalid, or becomes invalid during the search. This occurs if the mainframe disconnects for some reason.

Host Get Screen Dimensions

This Host Interface tool returns the screen dimensions for a given host connection.

Inputs

Host Connection

The name of the connection created by the [Host Connect](#) or the [Host Connect Ex](#) tool.

Outputs

Screen Width

The number of columns in the screen.

Screen Height

The number of rows in the screen.

Exit Paths:

Success

This path is taken if the screen dimensions are successfully retrieved.

Failure

This path is taken if the operation fails.

Host Get String

This Host Interface tool retrieves a value from a screen starting at a row and column you specify. You also specify the number of characters.

Inputs

Host Connection

The name of the connection created by the [Host Connect](#) tool.

Row

The row coordinate where you want to begin retrieving.

Column

The column coordinate where you want to begin retrieving.

Number of characters

The number of characters to retrieve.

Trim trailing whitespace from result string option

Select this option if you want to remove any blank spaces from the end of the value retrieved.

Outputs

Result String

The value retrieved.

Exit Paths

Success

This step takes the Success exit path if the value retrieved is written to a string.

Failure

This step takes the Failure exit path if the host connection is invalid, or becomes invalid during the operation. This occurs if the mainframe disconnects for some reason.

Host Initialize Resource

This Host Interface tool creates and initializes an instance of a resource pool. On some systems you can use the same device name for all sessions. However, on other systems a given device name cannot be used concurrently; some of these systems allow an arbitrary device name, but some systems have a fixed pool of device names from which to select.

A resource pool is simply a list of strings. Once you initialize the pool with the resource list, they can pass the resource pool name to the HostConnectEX tool, and it will assign an unused resource from the pool. If no resources are available in the pool, the request will fail after the specifiable number of retries. When the connection is disconnected, the resource is released back to the pool.

Inputs

Resource Pool Name

The name of the resource pool to establish.

List Of String Resources

The list of strings defining the resources within the resource pool.

Exit Paths

Success

This path is taken if the resource pool is successfully created.

Failure

This path is taken if the operation fails.

Host Move To Position

This Host Interface tool moves the cursor to the screen row and column coordinates you specify. Use this tool when you are already on the desired screen.

Inputs

Host Connection

The name of the connection created by the [Host Connect](#) tool.

Row

The row coordinate for the desired cursor location.

Column

The column coordinate for the desired cursor location.

Exit Paths

Success

This step takes the Success exit path if the cursor is successfully moved.

Failure

This step takes the Failure exit path if the host connection is invalid, or becomes invalid during the operation. This occurs if the mainframe disconnects for some reason.

Host Move To Screen

This Host Interface tool tells the [Host Server](#) to navigate to the specified screen. The destination screen must be specified in your [host profile](#).

Inputs

Host Connection

The name of the connection created by the [Host Connect](#) tool.

Host Profile

The name of a valid [host profile](#) defined on the Host Server.

Beginning Screen

The name of the screen to start on.

Destination Screen

The name of the screen to which the Host Server navigates.

Exit Paths

Success

This step takes the Success exit path if the destination screen is reached.

Failure

This step takes the Failure exit path if the host connection is invalid, or becomes invalid during the operation, or if the host profile is invalid. This occurs if the mainframe disconnects for some reason.

Host Press Key

This Host Interface tool types keystrokes into the terminal emulation. [For a list of valid keystrokes available for Mainframes and AS400s, click here.](#)

Inputs

Host Connection

The name of the connection created by the [Host Connect](#) tool.

Key to press

One or more keys that are typed into the screen.

Exit Paths

Success

This step takes the Success path if the keys are typed.

Failure

This step takes the Failure exit path if the host connection is invalid, or becomes invalid during the operation, or if the host profile is invalid. This occurs if the mainframe disconnects for some reason.

Host Put Form Data

This Host Interface tool writes information to a screen. The fields this tool writes to are defined in a [host profile](#).

Settings

Host Connection

The name of the connection created by the [Host Connect](#) tool.

Profile

The name of a valid [host profile](#) defined on the Host Server.

Screen Name

The name of the screen to on which to insert the data.

Field Bindings list

This list contains the field names defined for the specified screen. These fields are defined in your host profile.

Field

A field defined in the host profile.

Expression

An expression built with Expression Editor Assistant that defines the value to be typed into the field. To edit an expression, select an expression and then click the **Binding** button.

Binding button

Click this button to open the Expression Editor Assistant and assign an expression to a field.

Clear All Bindings

Deletes all expressions associated with the fields.

Exit Paths

Success

This step takes the Success exit path if the values are written to the form fields.

Failure

This step takes the Failure exit path if the host connection is invalid, or becomes invalid during the operation, or if the host profile is invalid. This occurs if the mainframe disconnects for some reason.

Host Put String

This Host Interface tool writes information to a screen at the row and column coordinate you specify. The fields this tool writes to are defined in a [host profile](#).

Inputs

Host Connection

The name of the connection created by the [Host Connect](#) tool.

Row

The row coordinate for the position where the string is written.

Column

The column coordinate for the position where the string is written.

String to be entered

The string value to write to the screen.

Exit Paths

Success

This step takes the Success exit path if the string is written to the screen.

Failure

This step takes the Failure exit path if the host connection is invalid, or becomes invalid during the operation. This occurs if the mainframe disconnects for some reason.

Host Verify Screen

This Host Interface tool verifies that you are on the expected screen. The [Host Server](#) uses a [host profile](#) to identify and move between screens. Each screen has a certain number of fields or identifying elements that allow Host Server to verify that it is on the specified screen. You might use this tool to verify that you are in the correct location after connecting via a cached Host Connection.

Inputs

Host Connection

The name of the connection created by the [Host Connect](#) tool.

Host Profile

The name of a valid [host profile](#) defined on the Host Server.

Screen

The name of a screen defined in your host profile.

Exit Paths

Success

This step takes the Success exit path if the screen is validated.

Failure

This step takes the Failure exit path if the screen is not the one defined in the Screen parameter.

Host Wait For Cursor

This Host Interface tool waits for the cursor to appear at the specified position.

Inputs

Host Connection

The handle for the connection to the host mainframe.

Row

The row coordinate to watch.

Column

The column coordinate to watch.

Timeout

The number of milliseconds to wait before taking the Timeout exit path.

Exit Paths

Success

This step takes the Success exit path if it detects the cursor at the specified position.

Failure

This step takes the Failure exit path if the host connection is invalid, or becomes invalid during the operation. This occurs if the mainframe disconnects for some reason.

Timeout

This step takes the Timeout exit path if the cursor is not detected before the Timeout value elapses.

Host Wait For ReadyToSend

This Host Interface tool pauses a handler until the mainframe sends a Ready To Send (RTS) signal to the Host Server. You can also specify that it wait for more than one RTS signal.

Inputs

Host Connection

The name of the connection created by the [Host Connect](#) tool.

Number of occurrences of ReadyToSend

The number of RTS signals to wait for before taking the Successful exit path.

Timeout (per RTS occurrence)

The number of milliseconds to wait before taking the Timeout exit path.

Exit Paths

Success

This step takes the Success exit path if it receives the correct occurrence of the RTS signal.

Failure

This step takes the Failure exit path if the host connection is invalid, or becomes invalid during the operation. This occurs if the mainframe disconnects for some reason.

Timeout

This step takes the Timeout exit path if an RTS signal is not received before the Timeout value elapses.

Host Wait for Screen

This Host Interface tool waits for a specified screen defined in the host profile to appear. You might use this to verify that you are on the correct screen before proceeding. This tool uses the validation rules specified in the host profile to verify the screen. See validation rules in the Interaction Host Recorder documentation for more information.

Inputs

Host Connection

The name of the connection created by the [Host Connect](#) tool.

Host Profile

The name of a published host profile. You may select from a list of published host profiles.

Screen

The name of a screen as defined in the host profile.

Timeout (milliseconds)

The number of milliseconds to wait before taking the Timeout path. Refer to the following table for more information:

Timeout Value	Affect
0	Immediate
>0	Use this value instead of the screen's default timeout.
<0	Use the screen default timeout.

Exit Paths

Success

This step takes the Success exit path if the string appears.

Failure

This step takes the Failure exit path if the host connection is invalid, or becomes invalid during the operation. This occurs if the mainframe disconnects for some reason.

Timeout

This step takes the Timeout exit path if the string does not appear before the Timeout value expires.

Host Wait For String

This Host Interface tool waits for a specified string to appear at specified row and column on the terminal emulation. You might use this to verify that a screen has finished painting before putting or retrieving values.

Inputs

Host Connection

The name of the connection created by the [Host Connect](#) tool.

Signal String

The string to wait for.

Row

The row coordinate to watch when waiting for the string.

Column

The column coordinate to watch when waiting for the string.

Timeout (milliseconds)

The number of milliseconds to wait before taking the Timeout path.

Case Sensitive search

Choose this option to match based on case. Deselect this option to ignore case.

Exit Paths

Success

This step takes the Success exit path if the string appears.

Failure

This step takes the Failure exit path if the host connection is invalid, or becomes invalid during the operation. This occurs if the mainframe disconnects for some reason.

Timeout

This step takes the Timeout exit path if the string does not appear before the Timeout value expires.

Icon

Overview of Interaction Conference (ICon) Tools

The ICon tool allows Interaction Conference to determine if a call is destined for a phone number that is being monitored for conferencing.

Click the following tool for more information about that tool:

[ICon Handle Call](#)

ICon Handle Call

This Interaction Conference tool notifies Interaction Conference of an incoming call and compares the dialed phone number to the set of defined phone numbers reserved for conferencing. It informs the caller of whether or not Interaction Conference is taking control of the call.

Inputs

Call Identifier

The call ID for the incoming call. This ID is used to look up the dialed number to determine if it is one of the phone numbers monitored for conferences.

Exit Paths

Success

This path is taken if Interaction Conference is taking control of the call.

Failure

This path is taken if Interaction Conference is not monitoring the dialed phone number. Normal call processing resumes.

Internet

Internet tools

Internet tools are for building handlers that interact with people over the Internet. These interactions can be through a web browser or through Internet chat. CIC has the ability to generate custom web pages and pop-up chat utilities for a person browsing your web site. For an overview of CIC's Internet capabilities, see the technical reference guide on *Web Services* in your documentation library.

Click one of the following tools for more information about that tool:

[Create Callback](#)

[Escape URL](#)

[Generate HTML](#)

[UnEscape URL](#)

Create Callback

This Internet tool creates a [Callback](#) session with a remote user from a request submitted from a web site.

Inputs

Callback User Name

Name of the user responding to the callback request.

Callback Phone Number

Phone number of the remote party.

Callback Name of Remote Party

Name of the sender of the callback request.

Callback Session Subject

Subject of the callback request. This is a single line field like the subject field of an email, where you enter a callback session subject line. You cannot force this text to wrap using \n or \r, and the length is limited.

Queue to be transferred to

Identifier for the queue to receive the callback session.

Outputs

Callback Session Identifier

Unique identifier for the callback session.

Exit Paths

Success

This path is taken if the callback session is successfully created.

Failure

This path is taken if the operation fails.

Escape URL

This Internet tool formats an argument to be appended to a URL by converting any spaces and/or special characters into their ASCII equivalents preceded by the % sign.

Example:

Consider the following URL:

```
http://thissite.com?arg1=test&arg2=this
```

Everything following the "?" is an argument that can be used by code (CGI, Java Script, servlet, VB script). In this format the "?","=" and "&" are special characters. Also, spaces are not allowed because a space signals the end of the arguments.

But, suppose arg1 was to be "this is a test? & how come?" instead of "test" as show above, resulting in the following URL:

```
http://thissite.com?arg1=this is a test? & how come?
```

This would cause a problem because the formatting of that argument is not consistent with URL formatting. Using this tool, the above argument would have all spaces and special characters converted into their ASCII codes, with the URL-formatting-compliant result being:

```
http://thissite.com?arg1=this%20is%20a%20test%3F%20%26%20how%20come%3f
```

Note: You must escape "this is a test? & how come" *before* you tack it on to the URL you are building. If you try to escape the entire URL it may escape characters that should not be escaped.

Input

Escape a string for a URL

This string is the argument to be converted.

Output

Return Value

This string is the converted argument to be attached to the URL.

Exit Path

Next

This tool always takes the Next exit path.

Generate HTML

This Internet tool step generates an HTML document for a person browsing your web site. On the Settings page you can bind String or List of String variables from a handler to [substitution fields](#) set up in an HTML template. These values appear in the web page generated for the person browsing your web site. See *Creating Handlers that Send Information to a Webpage* in the *Interaction Web Tools Developer's Guide* in the PureConnect Documentation Library.

Settings Page

Web Connection

This is the variable that contains the name of the web connection passed in by the [HTML Event initiator](#). The web connection is set by the CGI-server. You can send only one web page generated by a Generate HTML step to a connection. When this step is finished, the web connection information is discarded.

Template

This drop-down combo box contains a list of templates. These templates are created from HTML documents and may contain a list of substitution fields. Templates represent parsed HTML files that will be displayed for the person interacting with your web site. If the template contains substitution fields, you'll see a list of all the substitution fields. Choose the Manage button to create a new template from an existing HTML document. See *Creating Handlers that Send Information to a Webpage* in the *Interaction Web Tools Developer's Guide* in the PureConnect Documentation Library.

Note on HTML Templates: When you publish a handler containing a Generate HTML step, Interaction Designer puts the template onto the CIC server so it will be available when the handler runs. A copy of the template kept in the handler (ihd file). In cases where there is already an HTML template of the same name on the CIC server, Interaction Designer must decide whether to overwrite that template with the template contained in the ihd file.

Interaction Designer marks all templates with timestamps, which are generated when the template is first parsed (in the properties of the Generate HTML step when the user selects the HTML file). Interaction Designer uses these timestamps to determine if the server's version of the HTML template is different than the ihd file's. If the server template is older, then Interaction Designer overwrites it with the newer version from the ihd file. There are situations, though, where the template in the ihd file is older than the one on the server.

Manage button

The Manage button opens the HTML Template Registration dialog box. In this dialog box you can create a new template by entering information into two fields, Template Name and Input HTML Filename.

In Template Name, type a name for the template you want to create.

In Input HTML Filename, type or browse for the HTML document that contains the substitution fields.

When you click the OK button, CIC will parse the HTML file, and create a template containing a list of substitution fields. These substitution fields are displayed in the list on the Settings page. You can bind values from a handler or subroutine to these substitution fields. The template is stored in the Directory Services.

Substitution field

This list contains the names of any substitution fields in the template you chose. You can bind an expression or variable to a substitution field by selecting a substitution field and clicking the Bind Variable button. This opens the Bind Expression to Substitution Field dialog box where you can select a variable or build an expression to bind to the selected substitution field. The only variables and expression types you can bind are String and List of String. All other types must be converted to a String or List of String. Double-clicking a substitution field has the same effect as clicking the Bind Expression button.

Value

The value (expression or variable) associated with the Substitution Field.

Bind Expression button

Click this button after selecting a substitution field to open the Bind Expression to Substitution Field dialog box. In this dialog box, you can select a variable or build an expression to bind.

Unbind Expression button

Click this button after selecting a substitution field to unbind the bound variable or expression.

Exit Paths

Next

This step always takes the Next exit path.

UnEscape URL

This Internet tool is the inverse of [Escape URL](#). It converts an escaped argument back into its original form by replacing the ASCII codes with the characters they represent.

Input

UnEscape a string for a URL

This string is the escaped argument to be converted with this tool.

Output

Return Value

This string is the argument with any ASCII code replaced with the appropriate characters.

Exit Path

Next

This tool always takes the Next exit path.

IpNotes

Overview of IpNotes

The IpNotes tool set allows the handler author to store and retrieve named entities consisting of an arbitrary number of named attributes of any of the ID-defined types Integer, String, or Date/Time. Each attribute is a list of values of a particular type. An IpNote is available to all handlers on a given server and is accessed by name. IpNotes can be removed explicitly, or can be set to terminate automatically after a time interval specified at creation.

See the white paper, *A Guide to Interaction Multi-Site*, for a more detailed explanation of using IpNotes tools in CIC.

Click on a tool below to learn more about that tool:

[Create Note](#)

[Get Notes Attribute Date/Time](#)

[Get Notes Attribute Integer](#)

[Get Notes Attribute String](#)

[Remove Note](#)

[Update Notes Attribute Date/Time](#)

[Update Notes Attribute Integer](#)

[Update Notes Attribute String](#)

Create Note

This IpNotes tool adds a new note that is available to all handlers and is accessible by name. A note is a list of named lists of three different types: Integer, String, and Date/Time. These lists are called attributes and can have any number of items in them as long as they are all of the same type. There can be any number of attributes per note.

Inputs

Note Name

The name by which the newly created note will be accessible.

Lifetime (minutes), 0 = infinite

If a non-zero value is specified, the note will exist until the time expires. Otherwise, the note will exist until explicitly removed by the [Remove Note](#) tool.

Exit Paths

Success

This path is taken if the note is successfully created.

Failure

This path is taken if the operation fails.

Note Exists

This path is taken if the note name is already present in the list.

Get Notes Attribute Date/Time

This IpNotes tool retrieves the attribute specified by the attribute name from the note specified by the note name. The list returned contains zero or more Date/Times.

Inputs

Note Name.

The name of a note created by the [Create Note](#) tool or retrieved from a message by [Multi-Site Get Note](#). The note must exist or the "Unknown Note" exit path will be taken.

Attribute Name

The string value that was used to create the attribute using the "overwrite" operation of one of the various tools that assign Note Attributes.

Outputs

Last Update Time

The Date/Time of the last successful update to any attribute in the note.

Date Time Value List

The list of values retrieved from the note.

Exit Paths

Success

This path is taken if the specified note's Date/Time is successfully retrieved.

Failure

This path is taken if the operation fails due to an unspecified error.

Unknown Note

This path is taken if the specified Note cannot be found.

Unknown Attribute

This path is taken if the specified attribute cannot be found.

Wrong Type

This path is taken if the specified attribute exists, but it is not a Date/Time.

Get Notes Attribute Integer

This IpNotes tool retrieves the attribute specified by the attribute name from the note specified by the note name. The list returned contains zero or more Integers.

Inputs

Note Name

The name of a note created by the [Create Note](#) tool or retrieved from a message by [Multi-Site Get Note](#). The note must exist or the "Unknown Note" exit path will be taken.

Attribute Name

The string value that was used to create the attribute using the "overwrite" operation of one of the various tools that assign Note Attributes.

Outputs

Last Update Time

The Date/Time of the last successful update to any attribute in the note.

Integer Value List

The list of vales retrieved from the note.

Exit Paths

Success

This path is taken if the specified note's Date/Time is successfully retrieved.

Failure

This path is taken if the operation fails due to an unspecified error.

Unknown Note

This path is taken if the specified Note cannot be found.

Unknown Attribute

This path is taken if the specified attribute cannot be found.

Wrong Type

This path is taken if the specified attribute exists, but it is not an integer.

Get Notes Attribute String

This IpNotes tool retrieves the attribute specified by the attribute name from the note specified by the note name. The list returned contains zero or more Strings.

Inputs

Note Name

The name of a note created by the [Create Note](#) tool or retrieved from a message by [Multi-Site Get Note](#). The note must exist or the "Unknown Note" exit path will be taken.

Attribute Name

The string value that was used to create the attribute using the "overwrite" operation of one of the various tools that assign Note Attributes.

Outputs

Last Update Time

The Date/Time of the last successful update to any attribute in the note.

String Value List

The list of values retrieved from the note.

Exit Paths

Success

This path is taken if the specified note's Date/Time is successfully retrieved.

Failure

This path is taken if the operation fails due to an unspecified error.

Unknown Note

This path is taken if the specified Note cannot be found.

Unknown Attribute

This path is taken if the specified attribute cannot be found.

Wrong Type

This path is taken if the specified attribute exists, but it is not a string.

Remove Note

This IpNotes tool removes a previously created note. If a note is removed by any handler, it will cease to exist for any handler that tries to retrieve it after it has been removed.

Inputs

Note Name

The name of a note created by the [Create Note](#) tool or retrieved from a message by [Multi-Site Get Note](#). The note must exist or the "Unknown Note" exit path will be taken.

Exit Paths

Success

This path is taken if the note is successfully removed.

Failure

This path is taken if the operation fails.

Unknown Note

This path is taken if the note name was not found in the list.

Update Notes Attribute Date/Time

This IpNotes tool writes a named list of date/time values to an existing note. The user can choose to overwrite the old values, add more values, or selectively remove values already present.

Inputs

Note Name

The name of a note created by the [Create Note](#) tool or retrieved from a message by [Multi-Site Get Note](#). The note must exist or the Unknown Note exit path will be taken.

Update Operation: 0 = append, 1 = overwrite, 2 = remove

The "append" operation adds the supplied list of values to the values already present in the attribute.

The "overwrite" operation replaces any existing values with the supplied list. Note that the entire list is replaced, not the individual values. This means that if there were ten old values and you supplied a list of five new values, the resulting attribute would contain only the five new values, not five new and five old.

The "remove" operation searches for values in the existing list that match those in the supplied list and removes the first occurrence of each that is found.

Attribute Name

A string value by which the attribute of this type will be (or already is) accessible. The attribute must already exist if the update operation is append or remove. If not, the Unknown Attribute Exit Path will be taken. To set the initial values in a newly-created attribute, you must select the overwrite operation.

Date/Time Value List

The list of values to be added or removed from the note.

Exit Paths

Success

This path is taken if the specified note's Date/Time is successfully retrieved.

Failure

This path is taken if the operation fails due to an unspecified error.

Unknown Note

This path is taken if the specified Note cannot be found. This path will only apply when the update operation is Append or Remove.

Unknown Attribute

This path is taken if the specified attribute cannot be found. This path will only apply when the update operation is Append or Remove.

Wrong Type

This path is taken if the specified attribute exists, but it is not a Date/Time.

Update Notes Attribute Integer

This IpNotes tool writes a named list of integer values to an existing note. The user can choose to overwrite the old values, add more values, or selectively remove values already present.

Inputs

Note Name

The name of a note created by the [Create Note](#) tool or retrieved from a message by [Multi-Site Get Note](#). The note must exist or the Unknown Note exit path will be taken.

Update Operation: 0 = append, 1 = overwrite, 2 = remove

The append operation adds the supplied list of values to the values already present in the attribute.

The overwrite operation replaces any existing values with the supplied list. Note that the entire list is replaced, not the individual values. This means that if there were ten old values and you supplied a list of five new values, the resulting attribute would contain only the five new values, not five new and five old.

The remove operation searches for values in the existing list that match those in the supplied list and removes the first occurrence of each that is found.

Attribute Name

A string value by which the attribute of this type will be (or already is) accessible. The attribute must already exist if the update operation is append or remove. If not, the Unknown Attribute exit path will be taken. To set the initial values in a newly-created attribute, you must select the overwrite operation.

Integer Value List

The list of values to be added or removed from the note.

Exit Paths

Success

This path is taken if the specified note's Date/Time is successfully retrieved.

Failure

This path is taken if the operation fails due to an unspecified error.

Unknown Note

This path is taken if the specified Note cannot be found. This path will only apply when the update operation is Append or Remove.

Unknown Attribute

This path is taken if the specified attribute cannot be found. This path will only apply when the update operation is Append or Remove.

Wrong Type

This path is taken if the specified attribute exists, but it is not an integer.

Update Notes Attribute String

This IpNotes tool writes a named list of string values to an existing note. The user can choose to overwrite the old values, add more values, or selectively remove values already present.

Inputs

Note Name

The name of a note created by the [Create Note](#) tool or retrieved from a message by [Multi-Site Get Note](#). The note must exist or the Unknown Note exit path will be taken.

Update Operation: 0 = append, 1 = overwrite, 2 = remove

The append operation adds the supplied list of values to the values already present in the attribute.

The overwrite operation replaces any existing values with the supplied list. Note that the entire list is replaced, not the individual values. This means that if there were ten old values and you supplied a list of five new values, the resulting attribute would contain only the five new values, not five new and five old.

The remove operation searches for values in the existing list that match those in the supplied list and removes the first occurrence of each that is found.

Attribute Name

A string value by which the attribute of this type will be (or already is) accessible. The attribute must already exist if the update operation is append or remove. If not, the Unknown Attribute exit path will be taken. To set the initial values in a newly-created attribute, you must select the overwrite operation.

String Value List

The list of values to be added or removed from the note.

Exit Paths

Success

This path is taken if the specified note's Date/Time is successfully retrieved.

Failure

This path is taken if the operation fails due to an unspecified error.

Unknown Note

This path is taken if the specified Note cannot be found. This path will only apply when the update operation is Append or Remove.

Unknown Attribute

This path is taken if the specified attribute cannot be found. This path will only apply when the update operation is Append or Remove.

Wrong Type

This path is taken if the specified attribute exists, but it is not a string.

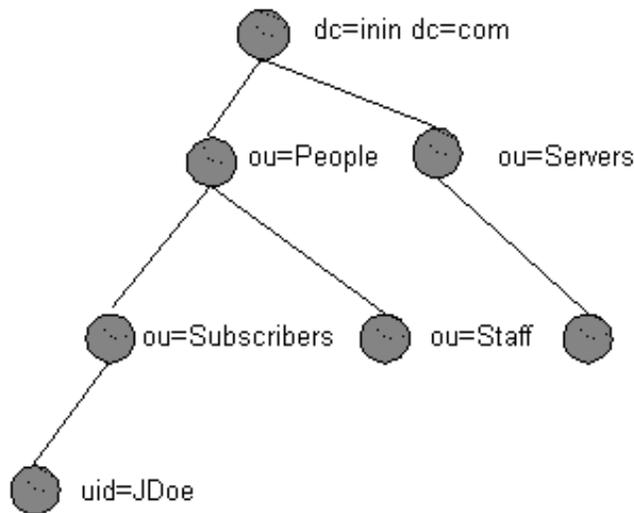
LDAP

Introduction to LDAP tools

LDAP (Lightweight Directory Access Protocol) is a protocol for accessing information in an information tree. The information tree is called a Directory Information Tree (DIT) and it is made up of parent and child directory entries. Each directory entry has a unique name, called a Distinguished Name (DN). Each entry also has a collection of attributes, which are name/value pairs that contain information about the entry.

The LDAP tools allow CIC to communicate with an LDAP server (LDAP v3 compliant) to retrieve attribute values from the directory entries. We foresee this as an alternative to retrieving user data from CIC's Directory Services. For example, a DIT might contain information about 10,000 users, including first name, last name, and email address. Instead of setting up CIC user accounts for each user, you can use the LDAP tools to look up the user email addresses from the LDAP server.

A sample DIT is shown in the following figure.

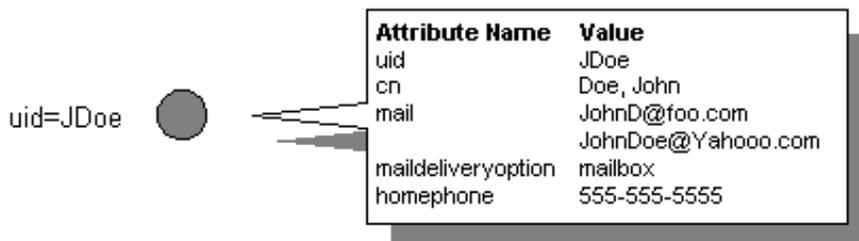


An entry's DN is comprised of its own name, and the names of its parent entries. For example, compare the JDoe entry DN to the previous figure to see how the DN is constructed:

```
uid=JDoe, ou=subscribers, ou=People, dc=inin, dc=com
```

Many of the LDAP tools require you to specify the DN you want to query or write to. Your DNs will look different since you have a different DIT. You should discuss your company's DIT structure with your LDAP administrator so that your handlers can construct valid DNs.

All entries contain attributes. Attributes are name/value pairs, and some attributes have more than one value, as illustrated in the following figure.



A person entry, such as JDoe in the previous figure, might contain a uid (user ID), common name (cn), mail, and many more attributes. The LDAP tools enable you to retrieve the value of an attribute in the directory entry you specify. Therefore, you could use the LDAP tools to search for all entries where the last name is Doe and return all email address attributes for those matching entries. Again, you'll need to discuss the entry types and their associated attributes with your LDAP administrator. He or she can give you a list of all entry types and their attribute names. You'll need this information when processing attributes within handlers.

The LDAP tools allow you to create and delete entries, and to read, write, and delete attribute values.

The Connection Cache

The LDAP tools internally maintain a connection cache of recently used LDAP connections. Each time an LDAP session is created, the [Session](#) tool first looks at the connection cache to see if there are any available connections for the specified server host name/login-information combination. If none are found, then a new connection is created with the specified LDAP server and that connection is added to the connection cache. When the handler that created that session ends, or when the [Close Session](#) tool is explicitly called for that session, the connection will be held in the connection cache for re-use by other handlers. A new connection will be created each time the Session tool is called and there are no available connections in the cache. There is no upper limit to the number of connections that will be held in the connection cache unless one is set by means of the [Set Max Cache Size](#) tool.

As an example of how this works, let's assume that no connections have been made to an LDAP server and that the connection cache is empty:

- 1) An LDAP session is created via the LDAP Session tool. The Session tool first looks in the connection cache to see if there are any available connections for the specified server host name/login-info combination. Finding none, a new connection is made with the specified LDAP server and added to the connection cache.
- 2) A second handler calls the LDAP Session tool for a second session. This second session tool also looks in the cache to see if an existing connection is available. It finds one connection there, but that connection is still in use. Since the first session is still being used (we didn't close it, and the handler is still running), then a second connection is created and added to the connection cache.
- 3) The handler that created session 1 then exits. The session is closed and the connection is marked as available in the connection cache. The actual physical connection with the LDAP server is still active at this point, though it is not in use.
- 4) A third handler then wants a connection to the same server (with the same login info). The session tool looks in the connection cache and sees two sessions. One that's still in use (by the second handler in step 2), and one that's available (the session created from the first handler that has now exited). Instead of having to make a new physical LDAP connection to the LDAP server, it just re-uses the existing LDAP connection established earlier (in step 1), marks it as in use in the cache and returns it to the session tool. Because the physical connection already existed, the new session was connected more or less instantly.

By default, the connection cache has no set limit on how many connections it will cache. An upper limit can be set and adjusted using the [Set Max Cache Size](#) tool. This sets the maximum number of connections that will be cached. Note that it does not set the maximum number of connections that can be made, just the number that will be stored in the cache. Other connections made above a set limit will be terminated when no longer in use.

For example, let's say we have 5 in-use connections and a maximum cache size is 5. If another tool needs another session and those other 5 connections are still in use, then a new session will still be created. New connections will never be refused. However, this sixth session object won't be added to the connection cache for later re-use because the maximum cache size has already been reached. When this sixth session handle goes away or is closed, then the LDAP connection will be immediately closed.

The LDAP Tools

This section briefly describes the tools and provides links to each tool's specific help topic. See the diagram in [LDAP: The order in which LDAP tools might be used](#) for more information on using these tools and the order in which they should be used.

[Add Entry](#)

Creates a new entry at the location you specify. You may also specify the entry's attributes and their values.

[Add Entry Ex](#)

Add Entry Ex creates a new entry like the Add Entry tool, but allows you to use an operation list to assign multiple values to an attribute.

[Add Operation](#)

Creates an LDAP operation and adds it to the operation list for execution with the Add Entry Ex or Modify Entry Ex tools. Operation lists are useful in LDAP because they allow you to perform multiple operations on multiple attributes over one connection to the LDAP server. For example, a caller might choose to modify their password and change several other personal settings through an IVR. The handler could add all of these operations to a list, and then execute them. You must specify an integer for the type of operation to perform.

Note: To create a list of multiple operations will probably use many instances of this tool in a handler, or create a loop so that this tool executes several times.

[Close Session](#)

The Close Session tool explicitly releases an LDAP session, making it available again in the cache.

[Delete Entry](#)

Deletes the specified LDAP entry (and all of its attributes).

[Flush Cache](#)

This tool removes all connections from the connection cache that are not currently in use.

[Get Cache Size](#)

Retrieves the number of connections presently in the connection cache.

[Get Entry Attributes](#)

Get Entry Attributes takes an entry handle (generated by the Read Entry or Next Entry tool) and extracts the value of an attribute you specify. The value must be one of the values you requested with the Read Entry or Search Entry tools. The attribute values are placed in a list of string variables. This is because an attribute can have more than one value.

[Login](#)

The Login tool logs a user into an LDAP server. The Login step must execute once in the handler before other LDAP tools are executed. You must specify a Login ID and Password. The other LDAP tools will have permissions based on that user's permissions. For example, if the tool logs in as administrator, the handler might have read/write access to the entire DIT. An individual user's login might grant access to only that user's entry.

[Modify Entry](#)

Changes the value of one or more [LDAP](#) entry attributes. Note that you may only specify one value for each attribute. However, if an attribute already has a value, you may use this tool to specify one additional value.

[Modify Entry Ex](#)

While the Modify Entry tool allows you to modify a single attribute, the Modify Entry Ex tool allows you to modify multiple attributes and to assign multiple values to a single attribute. This is useful because it reduces the network resources required to execute multiple operations. The operations list that this tool executes includes add, delete, and modify entry operations. Use the Add operation tool to create an operation and add it to an operation list. You might use Modify Entry Ex in a handler when a caller selects several configuration options in a single IVR session. Each configuration option could be stored in the handler's operation list and executed simultaneously before the handler ends.

[Move Entry](#)

Changes the RDN attribute and changes the parent. All of the attribute values remain unchanged except for the UID attribute.

[Next Entry](#)

The Search Entries tool generates the search result object that may contain multiple matching entries. There is an iterator in the search result object that points to one of those entries. The Next Entry tool generates a handle to the entry that the iterator is pointing to, and then moves the iterator to the next entry in the search result object. The entry handle can then be passed to the Get Entry Attribute tool (just like the entry handle the Read Entry tool creates. When all of the entries have been read, this tool takes the End of List exit path.

[Read Entry](#)

Read Entry retrieves specified attributes from a specified DS entry. It then creates an entry handle to that entry that you can pass to the Get Entry Attribute tool to extract the attribute value(s).

Rename Entry

Changes the RDN attribute of an LDAP entry without changing its parent. All of the attribute values remain unchanged except for the UID attribute.

Search Entries

Searches a specified portion of a DIT for entries that match one or more search criteria. The search criterion consists of attribute/value pairs and allows several types of wildcard matching. You also specify the attribute/value pairs you want to return for all matching entries. The matches are placed in a search result object.

Session

Session uses the Login ID to generate a session with the LDAP server. Several other LDAP tools that add, modify, and delete entries use the Session ID this tool generates. The session tool typically follows the Login tool.

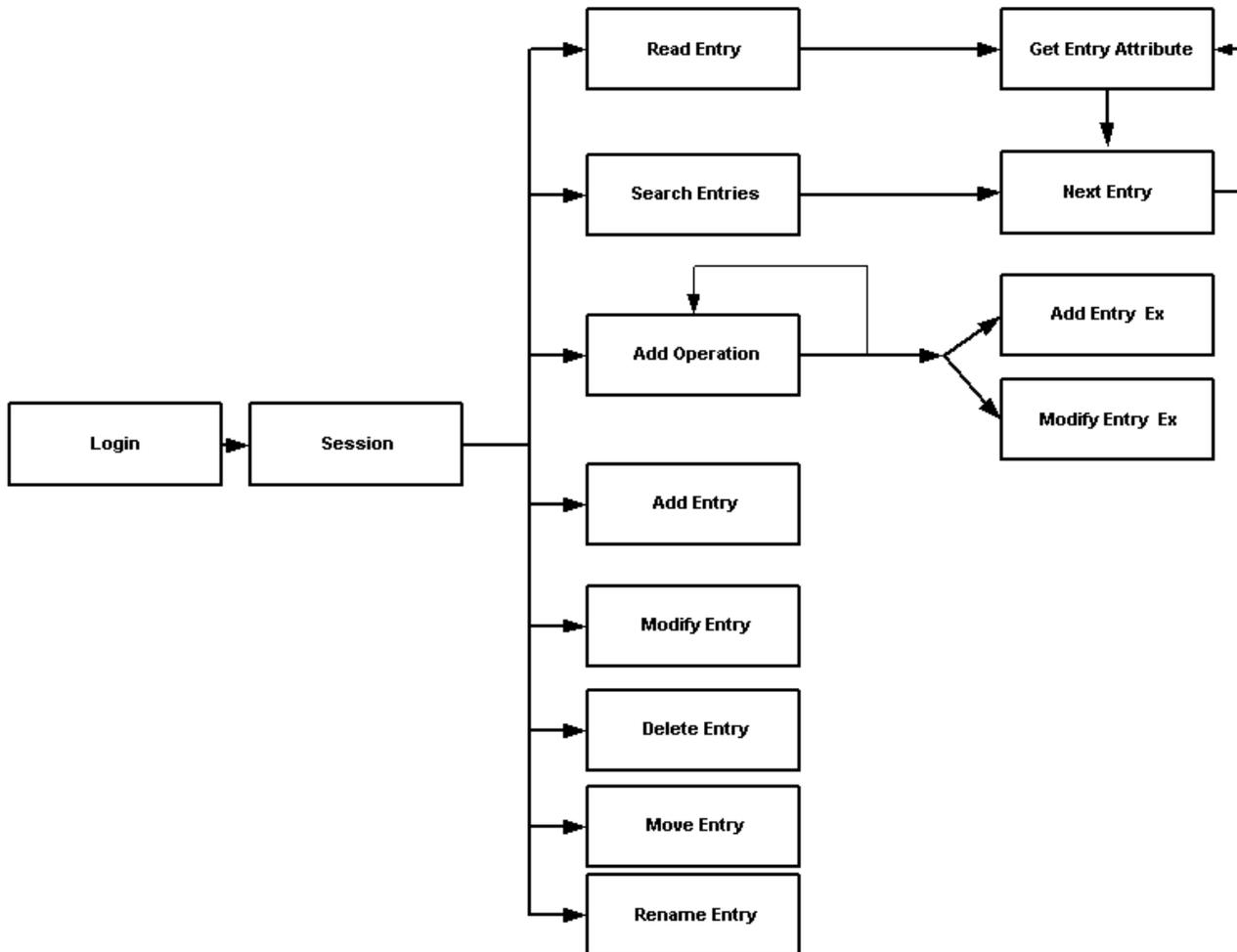
Set Max Cache Size

Sets an upper limit to the number of connections that will be held in the connection cache.

LDAP: The order in which LDAP tools might be used

This section illustrates the order in which the tools might be used. Tools like Delete Entry only require Login and Session, but other LDAP tools must precede Get Entry Attribute. This chart does not illustrate the non-LDAP tools you might use to perform such operations as collecting input from users or creating lists with attribute values.

See the [Introduction to LDAP Tools](#) for LDAP tool overview and descriptions.



Add Blob Attribute

This LDAP tool adds an operation into the list of operations to be executed on the LDAP server. The operation added is an operation to store the content of a blob (located in a file) as the value of an attribute on an entry of the LDAP server.

Inputs

Operation List

The operation list where the attribute will be added.

Attribute

The name of the attribute that is storing the blob.

Blob file

The file containing the content of the blob (icon, program, image, sound, prompt, etc.).

Outputs

Operation List

Operation list where the attribute was added.

Error Code

The error code (value)

Error Message

The error code name (string)

Error Native

The error returned by the native API (string)

Exit Paths

Success

This path is taken if the list of operations is successfully updated.

Failure

This path is take if the operation fails.

Add Entry

This LDAP tool adds an entry to the [LDAP](#) DS tree. You can only add entries if the user logged in with the [Login](#) tool has rights to add an entry. The DS schema may also restrict the types of entries that may be added. See [Introduction to LDAP tools](#) for more information on using this tool.

Inputs

Session Id

The handle to the session obtained with the [Session](#) tool.

EntryDN

The location within the LDAP DS where this entry is added. For example, you might use: "uid=JDoe, ou=subscribers, ou=People, dc=inin, dc=com". Contact your LDAP administrator for information regarding the entry location syntax for your LDAP directory information tree.

Attributes

A list of string value containing the attribute names to add with this entry. You might create this list using the [Parse String](#) tool.

Values

A list of strings (parallel to Attributes list) that contains values corresponding to the attribute names. You might create this list using the [Parse String](#) tool.

Note: Each attribute name may have only one value associated with it. If you want to assign more than one value to an attribute, use the Add Entry Ex tool.

Outputs

Error Code

The code associated with a processing error.

Error Message

A string description describing the error that occurred.

Error Native

A string description of an error that occurred in the LDAPLib module that connects this CIC server to your LDAP Server.

Exit Paths

Success

This step takes the Success of the operation completed successfully.

Failure

This tool takes the Failure exit path if an error occurred. Examine the output error text to determine the problem.

Add Entry Ex

This LDAP tool creates an entry at the location you specify with the attributes you specify in the operations. This tool allows you to create entry attributes with multiple values. If each attribute in the entry needs only one value, then use the [Add Entry](#) tool. See [Introduction to LDAP tools](#) for more information on using this tool.

Note: The operations list that this tool requires may contain only Add operations. See [the Add Operation](#) tool documentation for more information.

Inputs

Session Id

The session identifier generated with the [Session](#) tool.

EntryDN

The distinguished name (DN) of the entry this tool creates. For example, you might use: "uid=JDoe, ou=subscribers, ou=People, dc=inin, dc=com". Contact your LDAP administrator for information regarding the entry location syntax for your LDAP directory information tree.

OperationList

A list of operations to execute. This operation list is generated with one or [Add Operation](#) tools.

Outputs

Error Code

The code associated with a processing error.

Error Message

A string description describing the error that occurred.

Error Native

A string description of an error that occurred in the LDAPLib module that connects this CIC server to your LDAP Server.

Exit Paths

Success

This step takes the Success of the operation completed successfully.

Failure

This tool takes the Failure exit path if an error occurred. Examine the output error text to determine the problem.

Add Operation

This LDAP tool creates an LDAP operation and adds it to the operation list for execution with the [Add Entry Ex](#) or [Modify Entry Ex](#) tools. Operation lists are useful in LDAP because they allow you to perform multiple operations on multiple [LDAP](#) entries over one connection. For example, a caller might choose to modify their password and change several other personal settings through an IVR. The handler could add all of these operations to a list, and then execute them. You must specify an integer for the type of

operation to perform. You must also specify an attribute to operate on and optionally an attribute value to use (with add and modify operations.) See [Introduction to LDAP tools](#) for more information on using this tool.

Note: If you are creating an operation list for the Add Entry Ex tool, you may only specify Add operations. This is because Add Entry Ex should only be used to created entries that contain multi-valued attributes.

Inputs

Operation List

The operation list to which the operation should be added. If you do not specify an operation list, one is created for you. The operation name is then available from the output Operation List parameter.

Operation

The type of operation to add to the operation list.

Note: If you are creating an operation list for the Add Entry Ex tool, you may only specify Add operations. This is because Add Entry Ex should only be used to created entries that contain multi-valued attributes.

Refer to the following table:

0=Delete Attribute. If the logged in user has permissions, this operation deletes all values for the specified attribute. Do not use this operation in an operations list created for the [Add Entry Ex](#) tool.

1=Add Attribute. If the logged in user has permissions, this operation adds another value to the attribute. This operation allows you to create multi-valued attributes. For example, if the attribute foo contains the values "A" and you add the value "C," foo's value will be "A" and "C" after this operation.

2=Replace attribute replaces one or more existing attribute values with the attribute value that you specify. For example, if the attribute foo contains the values "A" and "B" and you replace them with "C", foos value will be "C" after this operation. Do not use this operation in an operations list created for the [Add Entry Ex](#) tool.

Attribute

Specify the attribute to operate on.

Values

For Add and Modify operations, specify the attribute values to use when changing the entry. For delete operations you should not specify a value.

Outputs

OperationList

If you did not specify the name of an operation list in the Operations List input parameter, this is the name of the operation list that was created for you. You will need this operation list when executing the list with the Add Entry Ex or Modify Entry Ex tools.

Error Code

The code associated with a processing error.

Error Message

A string description describing the error that occurred.

Error Native

A string description of an error that occurred in the LDAPLib module that connects this CIC server to your LDAP Server.

Exit Paths

Success

This step takes the Success of the operation completed successfully.

Failure

This tool takes the Failure exit path if an error occurred. Examine the output error text to determine the problem.

Close Session

This LDAP tool explicitly closes a single LDAP session, making that session available again in the cache.

Input

Session Id

The unique identifier for this session. This session ID is used by other LDAP tools.

Outputs

Error Code

The code associated with a processing error.

Error Message

A string description describing the error that occurred.

Error Native

A string description of an error that occurred in the LDAPLib module that connects this CIC server to your LDAP Server.

Exit Paths

Success

This path is taken if the session is successfully closed.

Failure

This path is taken if the operation fails.

Delete Entry

This LDAP tool deletes an entry in a [LDAP](#) DS tree. See [Introduction to LDAP tools](#) for more information on using this tool.

Inputs

Session ID

The unique identifier for the session created with the [Session](#) tool.

EntryDN

The distinguished name (DN) of the entry this tool will delete. For example, you might use: "uid=JDoe, ou=subscribers, ou=People, dc=inin, dc=com". Contact your LDAP administrator for information regarding the entry location syntax for your LDAP directory information tree.

Outputs

Error Code

The code associated with a processing error.

Error Message

A string description describing the error that occurred.

Error Native

A string description of an error that occurred in the LDAPLib module that connects this IC server to your LDAP Server.

Exit Paths

Success

This step takes the Success of the operation completed successfully.

Failure

This tool takes the Failure exit path if an error occurred. Examine the output error text to determine the problem.

Explode DN

This LDAP tool converts an LDAP distinguished name (DN) into its component parts. For example, the distinguished name "uid=JohnDoe,cn=users,dc=domain,dc=com" would tokenize into a string array of four component elements. Element 0: "uid=JohnDoe" Element 1: "cn=users" Element 2: "dc=domain" Element 3: "dc=com"

This works with any LDAP implementation.

Inputs

Distinguished Name

Enter a properly formatted LDAP distinguished name. For example, "uid=JohnDoe,cn=users,dc=domain,dc=com"

Outputs

Tokens

The output is a list of tokens that each contain a segment of the DN. For example,

Element 0: "uid=JohnDoe"

Element 1: "cn=users"

Element 2: "dc=domain"

Element 3: "dc=com"

Exit Paths

Success

This path is taken if the list of operations is successfully updated.

Failure

This path is take if the operation fails.

Flush Cache

This LDAP tool removes all LDAP connections that are currently not in use from the cache. This would be useful in purging the cache after occasional peaks in activity to free up server connections for other uses.

For example, a server may have a high maximum cache size set to facilitate a relatively short period of high activity each day. After this high activity period ends, the large cache size is no longer necessary and the connections stored in the cache are essentially being wasted. Calling this tool after such a period would make those connections available for other uses until the next high activity period.

Note that using this tool too often will negate the usefulness of caching connections at all.

Outputs

Error Code

The code associated with a processing error.

Error Message

A string description describing the error that occurred.

Error Native

A string description of an error that occurred in the LDAPLib module that connects this CIC server to your LDAP Server.

Exit Paths

Success

This path is taken if all unused connections are removed from the cache.

Failure

This path is taken if the operation fails.

Get Cache Size

This LDAP tool retrieves the number of connections currently held in the LDAP connection cache.

Outputs

Cache Size

The number of connections currently in the cache.

Error Code

The code associated with a processing error.

Error Message

A string description describing the error that occurred.

Error Native

A string description of an error that occurred in the LDAPLib module that connects this CIC server to your LDAP Server.

Exit Paths

Success

This path is taken if the number of connections is successfully retrieved.

Failure

This path is taken if the operation fails.

Get Entry Attribute

After retrieving all of an [LDAP](#) entry's attribute information with the Read Entry tool, use this LDAP tool to retrieve the list of values for a specific attribute in that entry. For example, after retrieving all of the attributes for the user John Doe with the Read Entry tool, use this tool to get a list of email address values in the mail attribute. See [Introduction to LDAP tools](#) for more information on using this tool.

Inputs

Entry

The handle to the entry that was created with the Read Entry tool's Entry output parameter.

Attribute

The attribute whose values you want to retrieve.

Outputs

EntryDN

The DN containing the attribute you are querying.

Values

A list of strings containing one or more values retrieved.

Error Code

The code associated with a processing error.

Error Message

A string description describing the error that occurred.

Error Native

A string description of an error that occurred in the LDAPLib module that connects this CIC server to your LDAP Server.

Exit Paths

Success

This step takes the Success of the operation completed successfully.

Failure

This tool takes the Failure exit path if an error occurred. Examine the output error text to determine the problem.

Get Entry Blob Attribute

This LDAP tool extracts the content of a blob (Binary Large Object) located in an attribute value and stores it in a file. The Entry object must be retrieved by applying a search operation on the LDAP server.

Inputs

Entry

The entry object that has been retrieved via the LDAP search request.

Attribute

The name of the attribute that is storing the blob.

Blob file

The file where the blob content will be stored.

Outputs

EntryDN

The fully DN name of the entry specified by the Entry input above.

Blob file

The full name of the blob file created to store the content of the blob attribute.

Error Code

The error code (value).

Error Message

The error code name (string).

Error Native

The error returned by the native API (string).

Exit Paths

Success

This path is taken if the blob attribute is successfully stored.

Failure

This path is taken if the operation fails.

Login

This LDAP tool logs a user into the [LDAP](#) server. The level of access granted by this login depends upon the access rights of the

user logging in. The Login ID this tool generates is used by the Session tool. See [Introduction to LDAP tools](#) for more information on using this tool.

Do not use this tool if you want to log in anonymously.

Inputs

User DN

The full distinguished name of the user. For example:

```
"uid=DoeJ,ou=mail users,ou=people,o=xyxy.com"
```

Password

The password associated with the USER DN.

Authentication

An integer value specifying the type of security to log in with. Refer to the following table:

Specify this integer...	To use this authentication method...
0 (default)	Simple
1	SSL (encrypted)
2	SASL (Windows only)

Outputs

LoginID

The unique login ID is a handle that other tools can use to perform operations over this login. The Session tool requires this LoginID.

Error Code

The code associated with a processing error.

Error Message

A string description describing the error that occurred.

Error Native

A string description of an error that occurred in the LDAPLib module that connects this CIC server to your LDAP Server. You can use this value for debugging purposes.

Exit Paths

Success

This step takes the Success exit path if the login completes successfully.

Failure

This tool takes the Failure exit path if an error occurred. Examine the output error text to determine the problem.

Modify Entry

This LDAP tool changes the value of one or more [LDAP](#) entry attributes. Note that you may only specify one value for each attribute. However, if an attribute already has a value, you may use this tool to specify one additional value. See [Introduction to LDAP tools](#) for more information on using this tool.

Inputs

Session ID

The unique identifier for the session created with the [Session](#) tool.

EntryDN

The distinguished name (DN) of the entry this tool will modify. For example, you might use: "uid=JDoe, ou=subscribers, ou=People, dc=inin, dc=com". Contact your LDAP administrator for information regarding the entry location syntax for your LDAP directory information tree.

Attribute

A list of strings containing the name of one or more attributes to modify. You might create this list using the [Parse String](#) tool.

Values

A list of strings (parallel to the Attribute list) that specifies the values for the attributes to modify.

Operation

- 0=Delete Attribute If the logged in user has permissions, this operation deletes all values for the specified attributes.
- 1=Add attribute If the logged in user has permissions, this operation adds another value to the attributes. This operation allows you to create multi-valued attributes. For example, if the attribute foo contains the values "A" and you add the value "C", foos value will be "A" and "C" after this operation.
- 2=Replace attribute replaces one or more existing attribute values with the attribute value that you specify. For example, if the attribute foo contains the values "A" and "B" and you replace them with "C", foos value will be "C" after this operation.

Outputs

Error Code

The code associated with a processing error.

Error Message

A string description describing the error that occurred.

Error Native

A string description of an error that occurred in the LDAPLib module that connects this CIC server to your LDAP Server.

Exit Paths

Success

This step takes the Success of the operation completed successfully.

Failure

This tool takes the Failure exit path if an error occurred. Examine the output error text to determine the problem.

Modify Entry Ex

This LDAP tool executes the operations in an operations list. Each operation in the list can delete, add to, or modify the values of an [LDAP](#) entry's attribute. See [Introduction to LDAP tools](#) for more information on using this tool.

Inputs

Session ID

The unique identifier for the session created with the [Session](#) tool.

EntryDN

The distinguished name (DN) of the entry this tool will operate on. For example, you might use: "uid=JDoe, ou=subscribers, ou=People, dc=inin, dc=com". Contact your LDAP administrator for information regarding the entry location syntax for your LDAP directory information tree.

OperationList

A list of operations generated with the [Add Operation](#) tool.

Outputs

Error Code

The code associated with a processing error.

Error Message

A string description describing the error that occurred.

Error Native

A string description of an error that occurred in the LDAPLib module that connects this CIC server to your LDAP Server.

Exit Paths

Success

This step takes the Success of the operation completed successfully.

Failure

This tool takes the Failure exit path if an error occurred. Examine the output error text to determine the problem.

Move Entry

This LDAP tool moves a [LDAP](#) Directory Services entry to another location within the DS tree. You specify the name of the entry to move, the entries new Relative Distinguished Name (RDN) to where it will be located after the move, and the entries Parent Distinguished Name (DN). See [Introduction to LDAP tools](#) for more information on using this tool.

Inputs

Session ID

The unique identifier for the session created with the [Session](#) tool.

EntryDN

The distinguished name (DN) of the entry this tool will operate on. For example, you might use: "uid=JDoe, ou=subscribers, ou=People, dc=inin, dc=com". Contact your LDAP administrator for information regarding the entry location syntax for your LDAP directory information tree.

NewEntryRDN

The relative distinguished name of the entry to be created. This, combined with NewParentRDN, specifies the new location. If the DN is:

"uid=JDoe, ou=subscribers, ou=People, dc=inin, dc=com", then the RDN of the original entry would be "uid=Jdoe". To specify the new RDN of Jsmith, you would use:
"uid-Jsmith" in this parameter.

NewParentDN

The parent entry of the new location. This, combined with NewEntryRDN, specifies the new location. For example, you might move the JSmith entry to the "ou=GoldCustomers" by specifying the following string in this parameter:

"ou=GoldCustomers, ou=People, dc=inin, dc=com"

When this tool executes, it results in the new entry:
"uid=Jsmith, ou=GoldCustomers, ou=People, dc=inin, dc=com"

Outputs

Error Code

The code associated with a processing error.

Error Message

A string description describing the error that occurred.

Error Native

A string description of an error that occurred in the LDAPLib module that connects this CIC server to your LDAP Server.

Exit Paths

Success

This step takes the Success of the operation completed successfully.

Failure

This tool takes the Failure exit path if an error occurred. Examine the output error text to determine the problem.

Next Entry

This LDAP tool reads an [LDAP](#) entry in a search result object (generated with the search tool) and moves the iterator to the next element in the search result object.

For example, the Search tool generates a search result object with 3 entries and initializes the iterator to point to the first element, as shown in the following table. The Search tool automatically places the iterator at the first element when it creates the list.

See [Introduction to LDAP tools](#) for more information on using this tool.

Search Result Object containing 3 elements.

Element #	Iterator
0	•
1	
2	

The Next Entry tool extracts the entry ID from the element to which the iterator is pointing and places it in the Entry output parameter. (You can then use the [Get Entry Attribute](#) tool to extract the value of one of that entry's attributes. See the Search tool documentation for more information on selecting the attribute values to return with the search.)

Now the next entry tool moves the iterator to the next element in the search result object, as shown in the following figure:

Element #	Iterator
0	
1	•
2	

The next time the Next Entry tool executes, it extracts the entry id from the element to which the iterator is pointing, then moves the iterator to the next element, as shown in the following figure:

Element #	Iterator
0	
1	
2	•

The next time the Next Entry tool executes, it see that the iterator is pointing to the last element in the list, extracts no information, and takes the End of List exit path.

Inputs

Entries

This is the handle (output by the Search tool) to the search result object containing the matching entries. You can pass this handle to the [Get Entry Attribute](#) tool.

Start Iteration?

False by default. Set this value to True to reset the iterator to point to the first element in the list. You would use this if you wanted to cycle through the list again. Set this value to false to not move the iterator from the element to which it currently points.

Outputs

Entry

A handle to the entry that was read from the search result object. You can pass this entry handle to the Get Entry Attribute tool.

Error Code

The code associated with a processing error.

Error Message

A string description describing the error that occurred.

Error Native

A string description of an error that occurred in the LDAPLib module that connects this CIC server to your LDAP Server.

Exit Paths

Success

This tool takes the Success path if the entry was correctly read and if the iterator was moved to the next entry in the list.

End of List

This tool takes the End of List exit path if the iterator was moved to the last element in the list.

Failure

This tool takes the Failure exit path if an error occurred. Examine the output error text to determine the problem.

Read Entry

This LDAP tool retrieves specified attributes from a specified [LDAP](#) Directory Services entry. It then creates a handle to that entry that you can pass to the [Get Entry Attribute](#) tool. See [Introduction to LDAP tools](#) for more information on using this tool.

Inputs

Session ID

The unique identifier for the session created with the [Session](#) tool.

EntryDN

The distinguished name (DN) of the entry this tool will operate on. For example, you might use: uid=JDoe, ou=subscribers, ou=People, dc=inin, dc=com

Contact your LDAP administrator for information regarding the entry location syntax for your LDAP directory information tree.

Attributes

A list of string value containing the names of the attributes you want to retrieve. Leave this field blank to retrieve all attributes. You might use the [Parse String](#) tool to generate the list of attributes.

Timeout

The amount of time to wait before taking the Failure exit path.

Outputs

Entry

The handle to the entry. You can pass this handle to the [Get Entry Attribute](#) tool to extract the values of one of the attributes this tool acquired.

Error Code

The code associated with a processing error.

Error Message

A string description describing the error that occurred.

Error Native

A string description of an error that occurred in the LDAPLib module that connects this CIC server to your LDAP Server.

Exit Paths

Success

This step takes the Success of the operation completed successfully.

Failure

This tool takes the Failure exit path if an error occurred. Examine the output error text to determine the problem.

Rename Entry

This LDAP tool changes the Distinguished Name of an [LDAP](#) entry without changing its parent entry. See [Introduction to LDAP tools](#) for more information on using this tool.

Inputs

Session ID

The unique identifier for the session created with the [Session](#) tool.

EntryDN

The distinguished name (DN) of the entry this tool will operate on. For example, you might type: "uid=JDoe, ou=subscribers, ou=People, dc=inin, dc=com"

NewEntryRDN

The new distinguished name of the entry. You do not need to type the entire DN, just the portion you want to change. For example, you might type: "uid=Jsmoe". This would create the DN: "uid=Jsmoe, ou=subscribers, ou=People, dc=inin, dc=com"

Outputs

Error Code

The code associated with a processing error.

Error Message

A string description describing the error that occurred.

Error Native

A string description of an error that occurred in the LDAPLib module that connects this CIC server to your LDAP Server.

Exit Paths

Success

This step takes the Success of the operation completed successfully.

Failure

This tool takes the Failure exit path if an error occurred. Examine the output error text to determine the problem.

Search Entries

This LDAP tool queries a specified portion of an [LDAP](#) Directory Information Tree (DIT) for all entries matching specified criteria. For each matching entry, Search Entries returns the entry ID and a list of attribute/value pairs that you request. This tool can return 0, 1, or many matches to your search criteria. All matches are placed in a search result object. You can use the [Next Entry](#) tool to iterate through this list of matching entries.

For example, suppose you want to return the email addresses for all entries that have "Schiller" in their UID attribute value. For each match, you want the entry's email address ("mail" attribute). The search creates a search result object containing 3 entries: SchillerA, SchillerAS, and SchillerTU. Each entry has a mail attribute/value pair. You can use the Next Entry tool to extract the first

entry ID and move the iterator to the next entry, then use the [Get Entry Attribute](#) tool to read the value of the mail attribute associated with the entry ID. In this way you can extract the email address values for each entry in the search result object.

See [Introduction to LDAP tools](#) for more information on using this tool.

Note: Do not use the [Login](#) tool after this tool if you want to log in anonymously.

Inputs

Session ID

The unique identifier for the session created with the [Session](#) tool.

SearchBase

The portion of the DS you want to search. You must specify a DS object with a full DN, such as "ou=subscribers, ou=People, dc=inin, dc=com".

SearchFilter

An expression that specifies the types of entries to return. Some common filters are described in the table below, but you should consult an LDAP reference. Also, see the topic [LDAP: The order in which LDAP tools might be used](#).

Here are some common search filters:

Syntax	Description
"(sn=smith)"	This filter matches entries with the sn (surname) attribute whose value is exactly smith.
"(sn=smith*)"	Matches entries where the sn attribute begins with smith, such as smithers.
"(sn=*smith)"	Matches entries where the sn attribute ends with smith, such as goldsmith.
"(sn=smi*th)"	Matches entries where the sn attribute begins with smi and ends with th, such as smi.
"(sn~=smith)"	Matches entries that sound like smith. This function is implemented differently depending upon your LDAP server and the language it employs.
"(sn<=smith)"	Matches entries where the sn attribute is less than or equal to smith lexicographically.

SearchScope

Indicates the area you want to search. This parameter accepts three integer values, as shown in the following table:

Integer Value	Meaning	Description
2	"subtree"	Indicates that you want to search the base and all subentries beneath, including the subentries of subentries. Basically, you want to search EVERYTHING at or below the specified SearchBase object.
1	"onelevel"	Indicates that you want to search only the immediate child subentries of the specified SearchBase object.
0	"base"	Indicates that you want to search only the specified SearchBase object.

Timeout

The maximum number of seconds the LDAP server should spend trying to honor the search request. Specify 0 for no timeout at all.

SizeLimit

Specifies the maximum number of entries to retrieve. For example, if your size limit is 100, but the server locates 500 matching entries, this tool returns the first 100, and the result code LDAP_SIZELIMIT_EXCEEDED is returned. Set SizeLimit to 0 to return all matching entries, or the maximum number allowed by your server.

Attributes

A list of string containing the attributes you want to return for each matching entry. If you leave this field blank, all attributes are returned.

Outputs

Entries

The handle to the search result object. You can pass this handle to the Next Entry tool to extract the values of one entry for use with the [Get Entry Attribute](#) tool.

Number of Entries

The number of matching entries found.

Error Code

The code associated with a processing error.

Error Message

A string description describing the error that occurred.

Error Native

A string description of an error that occurred in the LDAPLib module that connects this CIC server to your LDAP Server.

Exit Paths

Success

This step takes the Success of the operation completed successfully.

Failure

This tool takes the Failure exit path if an error occurred. Examine the output error text to determine the problem.

Session

This LDAP tool generates a Session ID which is used other [LDAP](#) tools to send commands to the LDAP Server. See [Introduction to LDAP tools](#) for more information on using this tool.

Inputs

HostName

The name of the server hosting the LDAP DS (Directory Server). This value is not required if the host server is on the local network. You may specify the name of the server or that server's IP address.

SearchBase

The OU (Organization Unit) of the DS tree you want to search. For example, in the following DN (Distinguished Name) `uid=DoeJ,ou=mail users,ou=people,o=xyxy.com`, the OU is `mail users`.

Login Id

The login ID generated with the Login tool in a step preceding this one in the handler.

Host Port

This optional field specifies the port through which to connect to the LDAP server. The default is 0.

Timeout

This optional field is the number of seconds to wait before timing out (Failure exit). The default is 3 seconds.

SizeLimit

A limit to the number of times that can be returned with any queries made during this session. Queries return lists, and this setting limits the size of those lists for all queries in this session.

Outputs

V3Compliant

True if the LDAP server is LDAP version 3 compliant. False if the LDAP server is not LDAP version 3 compliant.

Session Id

The unique identifier for this session. This session ID is used by other LDAP tools.

Error Code

The code associated with a processing error.

Error Message

A string description describing the error that occurred.

Error Native

A string description of an error that occurred in the LDAPLib module that connects this CIC server to your LDAP Server.

Exit Paths

Success

This step takes the Success of the operation completed successfully.

Note: If unused connections are in the cache (see [Introduction to LDAP Tools](#) for information on the connection cache), then it is possible for this tool to take the Success exit path even if the LDAP server is not currently available. In this scenario, the next tool that attempts to perform an action using the cached session will fail.

Failure

This tool takes the Failure exit path if an error occurred. Examine the output error text to determine the problem.

Set Max Cache Size

This LDAP tool sets the maximum number of connections that can be held in the LDAP connection cache. This tool limits the number of connections that are stored for reuse later, but does not limit the total number of LDAP connections that can be made.

Inputs

Max. Cache Size

The maximum number of connections that will be stored in the cache.

Outputs

Error Code

The code associated with a processing error.

Error Message

A string description describing the error that occurred.

Error Native

A string description of an error that occurred in the LDAPLib module that connects this CIC server to your LDAP Server.

Exit Paths

Success

This path is taken if the maximum cache size is successfully set.

Failure

This path is taken if the operation fails.

List

List Tools

List tools access and sometimes modify the contents of a list variable. List variables are like regular variables, except that they can hold more than one value. For example, a variable of type ListOfString can contain two or more string values. List variables are created using the [InsertAtHead](#) and [InsertAtTail](#) tools.

List variables allow you to hold multiple values of a specific type. For example, a list variable of a string type could contain many string values. A list variable can be declared for each of the following data types: String, Integer, Boolean, Database, DateTime, DBConnection, and Numeric. There is no limit to the number of elements in a list.

For each element in a list variable is assigned a place number. The first place number is always zero, the second is one, the third is two, and so on. The first element in a list is called a Head. Therefore, the Head of a list is always at place number 0. The end of a list is called a Tail. The place number for a tail is always the number of the last element in a list. Therefore, the place number of the tail may change as elements are added or removed from a list.

You might want to use a list variable to store multiple items retrieved from multiple fetches to a database. For example, if you have a handler that allows callers to retrieve information on all financial transactions that have occurred for a specific account over the past five days, you could have a step that fetches information on all transactions and stores them in a list variable.

List variables may be created for the following data types:

- Accumulator Lock Key
- Boolean
- Call Id
- Conference Id
- Database
- Date Time
- DB Connection
- Diagnostic Data
- Fax Envelope Ids
- Fax File Ids
- Fax Object Ids
- Fax Page List Ids
- File Handle
- Host Interface Connections
- Integer
- Mail Folder
- MQConnection
- MQObject
- Numeric
- Prompt
- Recording
- Queue Period Statistics
- Recording Id
- Socket Handle
- String
- Web Connection

Click on one of the following tools for more information about that tool:

[Copy String List](#)

[Filter List](#)

[Find](#)

[GetAt](#)

[GetCount](#)

[InsertAt](#)

[InsertAtHead](#)

[InsertAtTail](#)

[List To String](#)

[Merge String Lists](#)

[Parse String](#)

[RemoveAll](#)

[RemoveAt](#)

[RemoveHead](#)

[RemoveTail](#)

[SetAt](#)

[Sort Lists](#)

[Update Data Pair Values](#)

Copy String List

This List tool makes a copy of a list of strings. You can also use it to rename a list of strings. The optional Boolean input, when set to True, allows for clearing the source list.

Inputs

Empty Source List

Set this Boolean parameter to True to empty the source list of strings, reducing the memory used to store the list. This is useful when using the tool to rename a list of string.

Outputs

Source String List

This is the source list of string that will be copied. This variable is passed by reference. This list may be cleared after copying by setting the Empty Source List input parameter to True.

Copied String List

This variable will contain an exact copy of the source string list. The order, case, number of elements, and any blank elements in the source list are preserved. No filtering of the list is performed.

Exit Paths

Next

This step always takes the Next exit path.

Filter List

This List tool removes entries that don't match the specified string in **String to be used for filter** parameter and removes the corresponding entries in up to four parallel lists. You can specify the criteria used in selecting the item.

Example 1: String to be used in Filter: "Dogs"

Position	List of Strings to Be Processed	Parallel List	Parallel List	Parallel List	Parallel List
0	Birds	Canary	Finch	Robin	Goose
1	Cats	Siamese	Tabby	Black	Persian
2	Dogs	Labrador	Beagle	Leonberger	Terrier
3	Lizards	Ghecko	Ghila	Monitor	Chameleon
4	Bears	Black	Brown	Polar	Koala

Resulting Lists:

Position	List of Strings to Be Processed	Parallel List	Parallel List	Parallel List	Parallel List
0	Dogs	Labrador	Beagle	Leonberger	Terrier

Example 2:

String to be used in Filter: "Dogs"

Retain entries with empty strings: true

Position	List of Strings to Be Processed	Parallel List	Parallel List	Parallel List	Parallel List
0	Birds	Canary	Finch	Robin	Goose
1	Cats	Siamese	Tabby	Black	Persian
2	Dogs	Labrador	Beagle	""	Terrier
3	Lizards	Ghecko	Ghila	Monitor	Chameleon
4	Bears	Black	Brown	Polar	Koala

Resulting Lists:

Position	List of Strings to Be Processed	Parallel List	Parallel List	Parallel List	Parallel List
0	Dogs	Labrador	Beagle	""	Terrier

Example 3:

String to be used in Filter: "Dogs"

Retain entries with empty strings: false

Position	List of Strings to Be Processed	Parallel List	Parallel List	Parallel List	Parallel List
0	Birds	Canary	Finch	Robin	Goose
1	Cats	Siamese	Tabby	Black	Persian
2	Dogs	Labrador	Beagle	""	Terrier
3	Lizards	Ghecko	Ghila	Monitor	Chameleon
4	Bears	Black	Brown	Polar	Koala

Resulting Lists:

Returns lists with no entries. All lists will be emptied if one of the elements in a corresponding matching entry is empty.

Example 4:

String to be used in Filter: "B"

Retain entries with empty strings: true

Perform leading substring comparison: true

Position	List of Strings to Be Processed	Parallel List	Parallel List	Parallel List	Parallel List
0	Birds	Canary	Finch	Robin	Goose
1	Cats	Siamese	Tabby	Black	Persian
2	Dogs	Labrador	Beagle	Leonberger	Terrier
3	Lizards	Ghecko	Ghila	Monitor	Chameleon
4	Bears	Black	Brown	Polar	Koala

Resulting Lists:

Position	List of Strings to Be Processed	Parallel List	Parallel List	Parallel List	Parallel List
0	Birds	Canary	Finch	Robin	Goose
1	Bears	Black	Brown	Polar	Koala

Inputs

String to be used for filter

The text string that is searched for in the List of Strings to be processed.

Perform leading substring comparison?

Set this parameter to false if you want to search for an exact match. Set this parameter to true if you want to match based on the first significant digits.

Perform case insensitive comparison?

True tells this tool to ignore case while searching for a match. False tells this tool to consider case when searching for a match.

Compare based on keypad mappings?

Keypad mappings are the letters associated with the numbers on a telephone keypad. These letters are printed on the keys of all telephones. By default this is set to false. Set this to true only in steps performing a lookup on text entered on the telephone keypad.

Retain entries with empty strings?

Set this parameter to false to remove entries with null values. Set this parameter to true to keep entries with null values.

Outputs

List of strings to be processed

The name of the variable that contains the list of strings to be processed.

Parallel list

The name of the variable that contains the second list of strings to be processed. This is a required field. If you do not have a second list, put the name of the first list here also.

Parallel list

The name of the variable that contains the third list of strings to be processed. This is a required field. If you do not have a third list, put the name of the first list here also.

Parallel list

The name of the variable that contains the fourth list of strings to be processed. This is a required field. If you do not have a fourth list, put the name of the first list here also.

Parallel list

The name of the variable that contains the fifth list of strings to be processed. This is a required field. If you do not have a fifth list, put the name of the first list here also.

Exit Paths**Next**

This step always takes the Next exit path.

Find

This List tool step searches a list variable for a specific value, starting the search at a specific position in the list. This step returns the position of the element. If the value is not found, the value of the output variable is not changed.

Settings Page

List

The list to be searched.

Value

The value to be searched for in the list.

Index

The position in the list at which to begin searching. Zero is the first position.

Output Variable

The integer variable to receive the first found element.

String Match Method (Available only for lists of strings)

This is the method that list of string values will be matched. The choices are:

- Full - The Value has to match entirely an entry in the given list
- Left - The Value has to match the left side of an entry in the given list
- Contains - The value has to be contained by an entry of the given list
- Right - The value has to match the right side of an entry in the given list

Case Sensitive Match (Available only for lists of strings)

Check this box to find match the case of the value in a list of strings.

Exit Paths

Found

The Success exit path is taken if the value was found in the list.

Not Found

The Failure exit path is taken if the value was not found in the list.

GetAt

This List tool step retrieves a value from a specified element in a specified list. The retrieved value is stored in the output variable.

Settings Page

List

The list variable from which the value is retrieved.

Index

The position in the list variable from which the value is retrieved. Zero is the first position in the list.

Output Variable

The integer variable to receive the value of the element retrieved.

Exit Paths

Next

This step always takes the Next exit path.

GetCount

This List tool step counts the number of elements in a list and stores that count in a variable.

Settings Page

List

The list variable to be counted.

Output Variable

The integer variable that will receive the value of the number of elements.

Exit Paths

Next

This step always takes the Next exit path.

InsertAt

This List tool step inserts a specified value at a specified position in a list variable. If there is currently a value at that position, that value is moved down one position. For example, if you insert a value at position four, and there is already an older value at position four, the older value moves to position five. All older values in positions after the inserted value would move down.

Setting Page

List

The list variable to have a value inserted.

Value

The value to insert into the list.

Index

The position in the list where the value will be inserted. Zero is the first position in the list.

Exit Paths

Next

This step always takes the Next exit path.

InsertAtHead

This List tool step inserts a value at the beginning (position zero) of a specified list. This can also be used to define a new list variable.

Settings Page

List

The list variable to have a value inserted. Select "New" from the drop-down list to create a new List variable.

Value

The value to insert into the first position.

Exit Paths

Next

This step always takes the Next exit path.

InsertAtTail

This List tool step inserts a value at the end of a specified list. This can also be used to define a new list variable.

Settings Page

List

The list variable to have a value inserted. Select **New** from the list to create a new List variable.

Value

The value to insert into the last position.

Exit Paths

Next

This step always takes the Next exit path.

List to String

This List tool converts a list of string value to a delimited string value.

Inputs

List of Strings

The list of values to separate.

Delimiter Character

One or more characters to use when separating values in the output string.

Outputs

Delimited String

The string containing the values from the list separated by the delimiter character(s).

Exit Paths

Next

This tool always takes the Next exit path.

Merge String Lists

This List tool merges two lists of strings and returns the appended list, the intersection of the two lists, or the union of the two lists. The optional Boolean input, when set to True, allows for clearing the source lists.

Note: This tool is useful when you have two lists to check and you want to avoid looping through each item in each list.

Inputs

Mode

The type of merge to perform:

Value	Mode	Description	Example		
			Source List 1	Source List 2	Merged List
0	Intersect	The merged list contains only the items that are included in both source lists.	1, 2, 3	1, 3, 4	1, 3
1	Union	The merged list contains one instance of each item in the source lists.	1, 2, 3	1, 3, 4	1, 2, 3, 4
2	Append	The merged list contains all items from both lists.	1, 2, 3	A, B, C	1, 2, 3, A, B, C

Empty source lists?

Set this Boolean parameter to True to empty the source lists of strings, reducing the memory used to store the lists. This parameter is optional and defaults to False.

Outputs

Source List 1

The first source list of strings to merge.

Source List 2

The second source list of strings to merge.

Merged List

The resulting merged list.

Exit Paths

Next

This step always takes the Next exit path.

Parse String

This List tool breaks a string into fields and places those fields as elements in a list of strings. You can sometimes use this tool as a replacement for a series of [Insert at Tail](#) steps.

The string "John Doe | 555-1212 | Indianapolis, IN" becomes a list of strings with the following elements:

```
Element 0 "John Doe "  
Element 1 " 555-1212 "  
Element 2 " Indianapolis, IN"
```

Inputs

String to Parse

The string to process with this tool.

Delimiter character(s)

The character(s) to use when determining where the string is divided. The delimiter character is removed from the resulting elements. For example, if you specify "|" as your delimiting character, "|" will not appear in any of the resulting list of string elements.

Outputs

List of Parsed Strings

The list of strings containing the divided elements.

Exit Paths

Next

This tool always takes the Success exit path.

RemoveAll

This List tool step removes all the values from a list, leaving it empty.

Settings Page

List

The list variable to have its contents removed.

Exit Paths

Next

This step always takes the Next exit path.

RemoveAt

This List tool step removes a value at a specified position in a list variable. All remaining elements that remain shift up one position to fill the gap.

Settings Page

List

The list variable from which the value is removed.

Index

The position in the list variable from which the value is removed.

Exit Paths

Next

This step always takes the Next exit path.

RemoveHead

This List tool step removes the value at the first position (position zero) of a list variable. All elements below this position shift up.

Settings Page

List

The list variable from which the first element is removed.

Exit Paths

Next

This step always takes the Next exit path.

RemoveTail

This List tool step removes the value at the last position of a list variable.

Settings Page

List

The list variable from which the last element is removed.

Exit Paths

Next

This step always takes the Next exit path.

SetAt

This List tool step replaces the value of a specified element of a list variable with a new value. This does not cause a change in the size of the list because an older value in the position of the insertion is overwritten.

Settings

List

The list variable into which you want to insert a value.

Value

The value to insert into the list.

Index

The position in the list where the value will be inserted. Zero is the first position in the list.

Exit Paths

Next

This step always takes the Next exit path.

Sort Lists

This List tool sorts a list lexicographically, and sorts the elements of the four other parallel lists according to the sorting of the first. In other words, if the elements of the first list are rearranged in XX way, the elements in the other four lists will be rearranged in XX way.

For example, if you pass the list ("1", "3", "2") and a parallel list of ("Z", "Y", "X"), the outputs should be ("1", "2", "3") and ("Z", "X", "Y"); that is, the elements in the parallel lists are reordered according to the reordering that is performed on the list to be sorted.

Note that if there is a list of items that have the same key ("1", "1", "1"), the associated lists might not be in the same order following the sort.

Inputs

Sort in Ascending Order

Set this parameter to true to sort the list in ascending order. Set this parameter to false to sort in descending order.

Outputs

List of strings to be processed

This first list contains the elements to be sorted.

Parallel list

The elements of this list are sorted according to the sorting of elements of the first list. If you do not have a list for this parameter, you can repeat the name of a list from another parameter on this page.

Parallel list

The elements of this list are sorted according to the sorting of elements of the first list. If you do not have a list for this parameter, you can repeat the name of a list from another parameter on this page.

Parallel list

The elements of this list are sorted according to the sorting of elements of the first list. If you do not have a list for this parameter, you can repeat the name of a list from another parameter on this page.

Parallel list

The elements of this list are sorted according the sorting of elements of the first list. If you do not have a list for this parameter, you can repeat the name of a list from another parameter on this page.

Exit Path

Next

This tool always takes the Next exit path.

Update Data Pair Values

This List tool updates name/value pairs in two parallel string lists. It is called many times during a TUI session and is intended for relatively short lists (100 elements or less). For large lists, use the [Find](#) tool.

Note: This tool is intended to replace a subroutine in the XML TUI interpreter.

The advantage of using parallel lists of strings is that in-line parameter expressions can be used in tools to access values referenced by name. For example, suppose you want to get the Quantity and pass it to a tool as an Integer. You can do this with no additional tools. Use the expression as follows in an Integer tool parameter:

```
toI(GetAt(ValueList, Find(NameList, "Quantity", 0)))
```

Inputs

Data Name

The the name of the data pair.

Data Value

The value for the data pair.

Outputs

Name List

The string list containing the updated data pair names.

Value List

The string list containing the updated data pair values.

Exit Paths

Next

This step always takes the Next exit path.

Monitoring (Remoco)

Get Process Information

This Monitoring tool gets generic process information about a specified CIC subsystem.

Inputs

IC Server

The name of the PC on which CIC is running.

IC Subsystem

The subsystem you want to query.

Valid subsystems include:

Subsystem Name	Unicode Name	Unicode Debug Name
ACCServer	ACCServeru	ACCServerud
ACDServer	ACDServeru	ACDServerud
AdminServer	AdminServeru	AdminServerud
AlertServer	AlertServeru	AlertServerud
ClientServices	ClientServicesu	ClientServicesud
DataManager	DataManageru	DataManagerud
DialerServer	DialerServeru	DialerServerud
DSServer	DSServeru	DSServerud
Exchange Import	ExchangeImportu	ExchangeImportud
FaxServer	FaxServeru	FaxServerud
HostServer	HostServeru	HostServerud
IP	IP	IP
IPDBServer	IPDBServeru	IPDBServerud
IPServer	IPServeru	IPServerud
Notes Import	NotesImportu	NotesImportud
Notifier	Notifieru	Notifierud
PagingServer	PagingServeru	PagingServerud
SMDIServer	SMDIServeru	SMDIServerud
StatServer	StatServeru	StatServerud
SwitchoverService	Switchoveru	Switchoverud
TSServer	TSServeru	TSServerud

Outputs

CPU Usage

CPU usage of the specified subsystem.

Memory Usage

Memory usage of the specified subsystem. In Kbytes.

Elapsed Time

Elapsed time of specified subsystem has been running.

Number of Threads

Number of threads of specified subsystem.

Virtual Memory

The current size, in kilobytes, of memory that the subsystem has allocated and that cannot be shared with other processes.

Process ID

A number that uniquely identifies a running process.

Handle Count

The total number of handles currently open by this subsystem. This number is equal to the sum of the handles currently open by each thread in the subsystem.

Exit Paths

Success

This tool takes the Success exit path if it retrieved data about subsystem.

Failure

This tool takes the Failure exit path if it could not retrieve data about subsystem.

Merge Log Event ID and Insertion Strings

This Monitoring tool merges an event message with its insertion strings.

Inputs

Log Event Message ID

ID of event message. This is the same ID that is displayed in Event Viewer.

Insertion Strings

The strings to be inserted into the event message.

Outputs

Merged Message

Merged message of event message and insertion strings. Should be similar to messages in Event Viewer.

Exit Paths

Success

This tool takes the Success exit path if this tool was able to merge.

Failure

This tool takes the Failure exit path if this tool was unable to merge.

Merge Log Event Messages

This Monitoring tool merges an event message with its insertion strings.

Inputs

Log Event Message

Event message to be merged. Similar to messages in Event Viewer. Contains insertion parameters.

Insertion Strings

Strings to be merged with event msg.

Outputs

Merged Message

Merged message of event messages and insertion strings. Should be similar to messages in Event Viewer

Exit Paths

Success

This tool takes the Success exit path if this tool was able to merge the message.

Failure

This tool takes the Failure exit path if this tool was unable to merge the message.

Restart IC Subsystem

This Monitoring tool restarts an CIC subsystem.

Inputs

IC Server

The name of the PC on which CIC is running.

IC Subsystem

The subsystem you want to start.

Valid subsystems include:

Subsystem Name	Unicode Name	Unicode Debug Name
ACCServer	ACCServeru	ACCServerud
ACDServer	ACDServeru	ACDServerud
AdminServer	AdminServeru	AdminServerud
AlertServer	AlertServeru	AlertServerud
ClientServices	ClientServicesu	ClientServicesud
DataManager	DataManageru	DataManagerud
DialerServer	DialerServeru	DialerServerud
DSServer	DSServeru	DSServerud
Exchange Import	Exchange Importu	Exchange Importud
FaxServer	FaxServeru	FaxServerud
HostServer	HostServeru	HostServerud
IP	Ipu	Ipuud
IPDBServer	IPDBServeru	IPDBServerud
IPServer	IPServeru	IPServerud
Notes Import	Notes Importu	Notes Importud
Notifier	Notifieru	Notifierud
PagingServer	PagingServeru	PagingServerud
SMDIServer	SMDIServeru	SMDIServerud
StatServer	StatServeru	StatServerud
SwitchoverService	SwitchoverServiceu	SwitchoverServiceud
TSServer	TSServeru	TSServerud

Exit Paths

Success

If the subsystem is restarted, this tool takes the success path.

Failure

If the subsystem or server cannot be found, this tool takes the failure path.

Stop IC Subsystem

This Monitoring tool terminates an CIC subsystem.

Inputs

IC Server

The name of the PC on which CIC is running.

IC Subsystem

The subsystem you want to stop.

Valid subsystems include:

Subsystem Name	Unicode Name	Unicode Debug Name
ACCServer	ACCServeru	ACCServerud
ACDServer	ACDServeru	ACDServerud
AdminServer	AdminServeru	AdminServerud
AlertServer	AlertServeru	AlertServerud
ClientServices	ClientServicesu	ClientServicesud
DataManager	DataManageru	DataManagerud
DialerServer	DialerServeru	DialerServerud
DSServer	DSServeru	DSServerud
Exchange Import	Exchange Importu	Exchange Importud
FaxServer	FaxServeru	FaxServerud
HostServer	HostServeru	HostServerud
IP	Ipu	Ipud
IPDBServer	IPDBServeru	IPDBServerud
IPServer	IPServeru	IPServerud
Notes Import	Notes Importu	Notes Importud
Notifier	Notifieru	Notifierud
PagingServer	PagingServeru	PagingServerud
SMDIServer	SMDIServeru	SMDIServerud
StatServer	StatServeru	StatServerud
SwitchoverService	SwitchoverServiceu	SwitchoverServiceud
TSServer	TSServeru	TSServerud

Exit Paths

Success

If the subsystem named in the CIC Subsystem field was stopped, this tool takes the Success exit path.

Failure

If the subsystem named in the CIC Subsystem field was not stopped, this tool takes the Failure exit path.

Multi-Site

Multi-Site Create Message

This Multi-Site tool creates a message that results in a message handle that will be supplied to other Multi-Site tools.

Outputs

Message Handle

The variable you want to use as the handle to the newly created message. The message handle is a value that is required as input for all tools that operate on the message.

Exit Paths

Success

This path is taken if the message was successfully created.

Failure

This path is taken if the operation fails.

Multi-Site Get Integer

This Multi-Site tool retrieves an integer from any message.

Inputs

Message Handle

Variable assigned as the handle for the message from which the integer is to be retrieved. This variable is selected from the list of all valid message handle variables known to the handler.

Outputs

Value

The output variable that contains the integer read from the message. This variable will only contain a valid value if the success exit path was taken.

Exit Paths

Success

This path is taken if the integer is successfully retrieved.

Failure

This path is taken if the operation fails.

Multi-Site Get Note

This Multi-Site tool reads an IpNote from a message.

Inputs

Note Name

Unlike the other message elements, the name of the note is integral to the note data. A note name is required so that the system can better check for an error in retrieving notes as well as guaranteeing that a sent note element will be named the same on multiple servers. The Note Name is a string.

Message Handle

The handle of the message from which the data element is to be read.

Exit Paths

Success

This path is taken if the IpNote is successfully retrieved.

Failure

This path is taken if the operation fails.

Note Name Mismatch

The tool takes this path if the Note Name provided isn't the same on multiple servers.

Multi-Site Get String

This Multi-Site tool retrieves a string from a message.

Inputs

Message Handle

The handle of the message from which the string is to be read.

Outputs

Value

The value of the string read from the message.

Exit Paths

Success

This path is taken if the string is successfully retrieved.

Failure

This path is taken if the operation fails.

Multi-Site Put Integer

This Multi-Site tool places an integer value into the message.

Inputs

Message Handle

A valid message handle obtained from [Multi-Site Create Message](#) or [Multi-Site Message Received](#).

Value

The integer value that you want to put into the message.

Exit Paths

Success

This path is taken if the integer is successfully placed.

Failure

This path is taken if the operation fails.

Multi-Site Put Note

This Multi-Site tool places a note into a message.

Inputs

Note Name

The name of a note that was created by the [Create Note](#) tool or retrieved by the [Multi-Site Get Note](#) tool.

Message Handle

A valid message handle obtained from [Multi-Site Create Message](#) or [Multi-Site Message Received](#).

Exit Paths

Success

This path is taken if the note is successfully placed.

Failure

This path is taken if the operation fails.

Multi-Site Put String

This Multi-Site tool allows a text string to be placed in a message.

Inputs

Message Handle

A valid message handle obtained from [Multi-Site Create Message](#) or [Multi-Site Message Received](#).

Value

The string value that you want to put into the message.

Exit Paths

Success

This path is taken if the string is successfully placed.

Failure

This path is taken if the operation fails.

Multi-Site Send Event

This Multi-Site tool allows you to send a message that does not require a response.

Inputs

Message Handle

A valid message handle obtained from [Multi-Site Create Message](#) or [Multi-Site Message Received](#).

Destination

The number that identifies the site you want the message sent to. This is the site ID as configured at each site in DS via IA. Although the value is a number, the input to the tool must be a string. An asterisk (*) can be used to broadcast a message.

Object ID

And identifier to use for the object ID that will be included with the incoming message notification on the receiving server.

Event ID

An identifier to use for the event ID that will be included with the incoming message notification on the receiving server.

Exit Paths

Success

This path is taken if the message is successfully sent.

Failure

This path is taken if the operation fails.

Unknown Destination

This path is taken if the destination site ID provided is not found.

Multi-Site Send Request

This Multi-Site tool waits for a response to be received or the specified time to expire before continuing execution of the handler via the appropriate exit path.

Inputs

Message Handle

A valid message handle obtained from [Multi-Site Create Message](#) or [Multi-Site Message Received](#).

Destination

The number that identifies the site you want the message sent to. This is the site ID as configured at each site in DS via IA. Although the value is a number, the input to the tool must be a string. Only one destination can be specified for a request.

Object ID

And identifier to use for the object ID that will be included with the incoming message notification on the receiving server.

Event ID

An identifier to use for the event ID that will be included with the incoming message notification on the receiving server.

Timeout

The number of milliseconds to wait for a response before continuing. If a response is not received within the specified period of time, a timeout exit path is taken.

Outputs

Response Message Handle

The variable to use as a handle for the response message. The handle is returned when a response is received. If an exit path other than success is taken, then the variable will not contain a valid value.

Exit Paths

Success

This path is taken if the request is successfully sent.

Timeout

This path is taken if a response is not received in the specified period of time.

Failure

This path is taken if the operation fails.

Unknown Destination

This path is taken if the destination site ID provided is not found.

Multi-Site Send Response

This Multi-Site tool sends a response to a request. By using the correlation IDs obtained from the initiator, the response message is automatically routed back to the correct handler that is waiting for it.

Inputs

Message Handle

A valid message handle obtained from [Multi-Site Create Message](#) or [Multi-Site Message Received](#). This should be different from the handle for the incoming message.

Response Correlation ID

This value should be passed directly from the initiator output which defaults to the same name. This value should not be altered by the handler.

Response Destination ID

This value should be passed directly from the initiator output which defaults to the same name. This value should not be altered by the handler.

Exit Paths

Success

This path is taken if the response is successfully sent.

Failure

This path is taken if the operation fails.

Unknown Destination

This path is taken if the destination site ID provided is not found.

OCR

Overview of OCR Tools

Important: The system no longer supports OCR tools.

The OCR tools are an extra set of tools that ship with CIC OCR. These tools provide some of the OCR functionality in CIC, although you must also install OCR and Export Servers before OCR will work correctly.

Click one of the tools below for more information on that tool:

[Export of OCR Files](#)

[Export Parser](#)

[OCR for I3Fax files](#)

[OCR for TIFF/PCX/DCX files](#)

[OCR Parser](#)

See Also

[OCR Installed](#)

Export for OCR file

Important: The system no longer supports OCR tools.

Converts OCR result file (created by the [OCR for TIFF/PCX/DCX files](#) tool) into one of several document formats. You may optionally choose to preface unrecognized words and characters with a special character, allowing easier search-and-replace editing later.

Inputs

OCR filename

The name of the OCR result file generated by the [OCR for TIFF/PCX/DCX files](#) tool. This value is empty by default.

Text format

An integer corresponding to one of the following document types:

1. AmiPro2030
2. ANSI
3. ASCII
4. DatabaseASCII
5. DCA
6. DecolASCII
7. EBCDIC
8. Excel
9. FrameMaker
10. Interleaf
11. Lotus123WK1
12. Lotus123WK3
13. Lotus123WK4
14. LotusManuscript2x
15. MicrosoftWord40
16. MicrosoftWord5x
17. MicrosoftWord60
18. Multimate33
19. MultimateAdv36
20. MultimateAdv37
21. OfficeWriter6x
22. PageMaker
23. PDAFormat
24. PFSFirstChoice20
25. PFSFirstChoice30
26. PFSPProfWrite2x
27. Quattro
28. RichTextFormat
29. SamnaWordIVPlus
30. Ventura
31. WindowsWrite3x
32. WordForWindows1x
33. WordForWindows2x
34. WordForWindows60
35. WordForWindows70
36. WordPerfect42
37. WordPerfect50

- 38. WordPerfect5152
- 39. WordPerfect60
- 40. WordStar50
- 41. WordStar55
- 42. WordStar60
- 43. WordStar70
- 44. WordStarWindows1x
- 45. XyWriteIIIPlus

Mark suspicious characters/words?

When the OCR engine parses this image, it can optionally mark words or characters it wasn't sure about with a special character. The special character is placed before each low-confidence word and character. Specify true to automatically preface low-confidence words and characters with a special character. This makes search-and-replace editing easier if you use a character that rarely appears in the document, such as "#" or "~". Specify false to not mark low-confidence words and characters. The resulting quality of the exported text is dependent upon the quality of the fax image.

Character Marker

The character that will preface low-confidence words and characters. Pick a character that appears rarely in your document, such as "#" or "~".

Text file name

The name of the document to be generated. Leave this empty ("") to generate a unique filename that will be passed to the **Name of the output text file** *output* parameter.

Outputs

Text file name

This should be the same name as you specified in the **Name of output text file** *input* parameter.

Error Code

If this tool fails, you can evaluate the reason for failure by examining Error Code and Error Message. Error codes and messages are listed in the Error Message parameter explanation below.

Error Message

A description of why this tool failed.

Error Code	Error Message	Explanation
-1	"Fatal error."	This tool has failed due to a larger failure within CIC. Your event log may contain diagnostic information about the failure. This must be diagnosed by technical support. You should never encounter this error.
0	"No error."	This tool executed successfully.
1	"OCR not enable"	The OCR Server is not running. Make sure the OCR Server is installed, running, and properly configured from within Interaction Administrator.
2	"No OCR result filename."	The file could not be created at the specified location. Make sure the filename and path are correct.
3	"Bad OCR result filename."	The OCR result file could not be created at the specified location.
4	"Bad text output filename."	The file could not be created at the specified location. Make sure the filename and path are correct.

Exit Paths

Success

This tool takes the Success exit path if the Error Code is 0.

Failure

This tool takes the Failure exit path if the error code is not 0.

Export Parser

Important: The system no longer supports OCR tools.

OCR Parser analyzes the <Export> configuration string that contains instructions for how the attached image should be processed. This <Export> string is specified by the sender in the body of the email, and may specify one or more of the following elements:

TextFormat={Word7, ANSI, TXT, etc.}	Specifies the type of file to create from the OCR result file. See TextFormat codes for OCR Export for a list of acceptable codes.
Marker={#, ~, or any other character}	Specifies the character with which to mark unrecognized words and characters.

Inputs

String to parse

When you specify the string containing the body of an email, this tool will find the portion of the string contained between <Export> and </Export>.

Stop Parsing String

The text within the body of the email that indicates the end of the configuration info and the beginning of the text. For example, many Exchange messages begin with "Original Message".

Delete it after

True to delete the OCR Result file after the export document is created. False to leave the OCR Result file attached to the email sent back to the user. The email sent back will also contain the export file.

Text format

The type of document to create in the export operation. ANSI is the default.

Mark suspicious characters/words

True to precede unrecognized characters and words with a special character. False to ignore unrecognized characters and words.

Character Marker

The character to use as a marker. Specify a character that is not likely to appear in the document, such as ~ or ^.

Outputs

Default

If the <Export> configuration was specified, this is false. If <Export> was not found, this is false and the default input parameters were used.

Delete it after

This value is set depending upon whether or not the OCR result file was deleted.

Text format

The format of the exported document.

Mark suspicious characters/words

This value is set depending upon whether or not suspicious characters were marked.

Character Marker

The character used to mark suspicious characters and words.

Exit Paths

Success

If the operation is successful, this tool takes the Success exit path.

Failure

This tool takes the Failure path if it was configured incorrectly.

OCR for I3Fax files

Important: The system no longer supports OCR tools.

Processes a fax (an I3Fax file with an .i3f extension) and stores the data in the fax file. The [Export for OCR files](#) tool can extract this data from the fax file and convert it to a text file (or one of several other document formats).

Inputs

Fax identifier

The identifier for the fax to process.

First page

In a multi-page image file, the first page to render. 1, the default value, is the minimum allowable value.

Last page

In a multi-page image file, the last page to render. Entering a 0 in this field renders all pages. A value greater than 1 specifies a range. If your range is greater than the number of pages in the file, this tool fails.

Perform dictionary checking? option

Select this option to use an additional dictionary when processing the fax. You can create your own custom dictionary in a text file with words separated with spaces and an extra carriage return following the last word in the text file. Clear this option to use only the OCR server's default dictionary.

Dictionary file name

The name of the dictionary to use. You should also specify a fully qualified path. If the path is not included, this tool looks for the dictionary file in the directory specified in the Work Path Server Parameter in Interaction Administrator. Cleared is the default for this tool.

Maintain your custom dictionaries with commonly used words or acronyms. Update this file regularly to add new words. This would improve OCR accuracy and processing time.

Perform language analysis

Select this option if you want the OCR Server to recognize characters unique within a language. For example, if you want to recognize the German umlaut character, you would select this option and specify German in the Language parameter below. Cleared is the default for this parameter.

Language

The language for which you want to recognize special characters. See the Perform language analysis example above.

Country

Some languages have characters unique to a certain country's dialect. This parameter specifies which country.

Outputs

Error Code

If this tool fails, you can evaluate the reason for failure by examining Error Code and Error Message. Error codes and messages are listed in the Error Message parameter explanation below.

Error Message

A description of why this tool failed.

Error Code	Error Message	Explanation
-1	"Fatal error."	This tool has failed due to a larger failure within CIC. Your event log may contain diagnostic information about the failure. This must be diagnosed by technical support. You should never encounter this error.
0	"No error."	This tool executed successfully.
1	"OCR not enable"	The OCR Server is not running. Make sure the OCR Server is installed, running, and properly configured from within Interaction Administrator.
2	"Bad fax identifier."	The fax identifier is no longer valid. Make sure the fax files are being created successfully in the fax handlers.
3	"No dictionary filename."	The custom dictionary is not in the specified location. Verify that your path and filename are correct, and that the dictionary is in the location specified.
4	"Bad dictionary filename."	The custom dictionary is not in the specified location. Verify that your path and filename are correct, and that the dictionary is in the location specified.
5	"OCR analysis already done."	The fax file has already been processed by this tool and the OCR data is present in the fax file and ready for further processing.

Exit Paths

Success

This tool takes the Success exit path if the Error Code is 0.

Failure

This tool takes the Failure exit path if the error code is not 0.

OCR for TIFF/PCX/DCX files

Important: The system no longer supports OCR tools.

Processes a [TIFF](#) or [PCX](#) and renders data in an OCR result file. The [Export for OCR files](#) tool can convert the OCR result file to a text file (or one of several other document formats).

Inputs

Image file name

The source [TIFF](#) or [PCX](#) file from which the OCR result file is created. You should also specify a fully qualified path. If the path is not included, this tool looks for the dictionary file in the directory specified in the Work Path Server Parameter in Interaction Administrator.

First page

In a multi-page image file, the first page to render. 1, the default value, is the minimum allowable value.

Last page

In a multi-page image file, the last page to render. Entering a 0 in this field renders all pages. A value greater than 1 specifies a range. If your range is greater than the number of pages in the file, this tool fails.

Perform dictionary checking? option

Select this option to use an additional dictionary when processing the file. You can create your own custom dictionary in a text file with words separated with spaces and an extra carriage return following the last word in the text file. Clear this option to use only the OCR server's default dictionary. Cleared is the default setting.

Dictionary file name

The name of the dictionary to use. You should also specify a fully qualified path. If the path is not included, this tool looks for the dictionary file in the directory specified in the Work Path Server Parameter in Interaction Administrator.

Perform language analysis

Select this option if you want the OCR Server to recognize characters unique within a language. For example, if you want to recognize the German umlaut character, you would select this option and specify German in the Language parameter below.

Language

The language for which you want to recognize special characters. See the Perform language analysis example above.

Country

Some languages have characters unique to a certain country's dialect. This parameter specifies which country.

Outputs

OCR file name

The result file containing the rendered OCR data. The [Export for OCR files](#) tool takes this filename as an input. You should also specify a fully qualified path. If the path is not included, this tool creates the OCR result file in the directory specified in the Work Path Server Parameter in Interaction Administrator.

Error Code

If this tool fails, you can evaluate the reason for failure by examining Error Code and Error Message. Error codes and messages are listed in the Error Message parameter explanation below.

Error Message

A description of why this tool failed.

Error Code	Error Message	Explanation
-1	"Fatal error."	This tool has failed due to a larger failure within CIC. Your event log may contain diagnostic information about the failure. This must be diagnosed by technical support. You should never encounter this error.
0	"No error."	This tool executed successfully.
1	"OCR not enable"	The OCR Server is not running. Make sure the OCR Server is installed, running, and properly configured from within Interaction Administrator.
2	"No image filename."	The image file does not exist at the specified location.
3	"Bad image filename."	The image file does not exist at the specified location.
4	"No dictionary filename."	The custom dictionary is not in the specified location. Verify that your path and filename are correct, and that the dictionary is in the location specified.
5	"Bad dictionary filename."	The custom dictionary is not in the specified location. Verify that your path and filename are correct, and that the dictionary is in the location specified.
5	"Bad OCR result filename."	The OCR result file could not be created at the specified location.

Exit Paths

Success

This tool takes the Success exit path if the Error Code is 0.

Failure

This tool takes the Failure exit path if the error code is not 0.

OCR Parser

Important: The system no longer supports OCR tools.

OCR Parser analyzes the <Reco> configuration string that contains instructions for how the attached image should be processed. This <Reco> string is specified by the sender in the body of the email, and may specify one or more of the following elements:

FirstPage={1...n	Specifies the first page to process. If you do not specify a first page, the file is processed up to the LastPage you specify.
LastPage={0...n.}	Specifies the last page to process. If LastPage has the same value as FirstPage, only one page is processed. If LastPage=0, all pages are processed. If you do not specify LastPage, all pages after FirstPage are processed.
Dictionary= {any_dictionary_name.txt}	Specifies the name of the dictionary you have created in \CIC\work\dictionary directory on your CIC Server. You must include the .TXT extension. Do not specify a path, and do not include spaces in your dictionary names.
Language= {English,French,German, British,Spanish,Italian,Swedish, Danish,Dutch,Norwegian,Brazilian, Portuguese,Finish}	Specifies the language to use when recognizing characters.

Inputs

String to parse

When you specify the string containing the body of an email, this tool will find the portion of the string contained between <Reco> and </Reco>.

Stop Parsing String

The text within the body of the email that indicates the end of the configuration info and the beginning of the text. For example, many Exchange messages begin with "Original Message". Delete the source file after

Delete the source file after

Specify true to include the source image in the reply. Specify false to remove the source image from the reply.

First Page

The first page to process in a multi-page tif file in the range of 1 to n.

Last Page

The last page to process in a multi-page .tif file in the range of 0 to n. Specify 0 to process all pages. Specify the same value as First page to process only one page.

Perform dictionary checking

True to use a special dictionary. False to use only the built in dictionary. OCR processing uses a built-in dictionary even if you specify False. This parameter is false by default.

Dictionary file name

Specify the name of a custom dictionary. The dictionary must be contained in the \EIC\work\dictionary directory on your CIC server. Dictionaries are text documents (like I3.txt with words separated by spaces or line breaks with the last word followed by a hard return. You must include the .TXT extension. Do not specify a path, and do not use spaces in your dictionary names. "" by default. You may specify only one custom dictionary.

Perform language analysis

True to turn on language analysis. False to turn off language analysis. False by default.

Language

Specify one of the following languages if Perform Language Analysis parameter is set to True:
English,French,German,British,Spanish,Italian,Swedish,Danish,Dutch,Norwegian,Brazilian,Portuguese, Finish. "English" by default.

Outputs

Default

If the <Reco> configuration was specified, this is false. If <Reco> was not found, this is false and the default input parameters were used.

Delete the source file after

Specify true to include the source image in the reply. Specify false to remove the source image from the reply. This value is set depending upon whether or not the source file was included in the output.

First Page

The first page that was processed.

Last Page

The last page that was processed.

Perform dictionary checking

This value is set depending upon whether or not dictionary checking was used.

Dictionary file name

The dictionary that was used.

Perform language analysis

This value is set depending upon whether or not language analysis was used.

Language

Indicates the language used to parse the fax for textual content. Specify one of the following languages if Perform Language Analysis parameter is set to True:
English,French,German,British,Spanish,Italian,Swedish,Danish,Dutch,Norwegian,Brazilian,Portuguese, Finish. "English" by default.

Country

The country that was used.

Exit Paths

Success

If the operation is successful, this tool takes the Success exit path.

Failure

This tool takes the Failure path if it was configured incorrectly.

Personal Rules

Overview of Personal Rules

Personal rules tools allow you to extend rule functionality to other events generated by CIC, and then determine the parameters of the action handles that are outputs of the rules.

Click one of the following tools for more information:

[Personal Action Details](#)

[Personal Rules](#)

Personal Action Details

Use this Personal Rules tool to get parameters from specific personal rules action handles. Personal rules action handles are outputs from the Personal Rule tool and contain a list of actions.

Inputs

Action Handle

The personal rule action handle that contains all the actions for a rule.

Action Index

This integer is the index of the action you want to get details about. The list has a "0" index, so if you want to get details about the first action, use index "0."

Outputs

Action

This string contains the name of the action.

Parameter List

This list of strings contains the parameters for the action.

Exit paths

Continue

This path continues on to the next step.

Abort

This step aborts the process.

Personal Rules

A personal rule automatically manages your interactions. For example, this is usually done through the Rules dialog box in Interaction Desktop. With this Personal Rules tool, you can extend the rule functionality to other events generated by CIC. The rules create a list of actions. You can then look up any action on the list to get its parameters.

Inputs

Call Id

The unique identifier for the interaction from which you want to get personal rules.

User Id

The fully qualified user ID of the user who has the rule.

Entry Point

This string value indicates the set of rules to process. For example, one entry point might be for incoming calls. Another set might be for incoming faxes. This allows you to have different rules for different interactions.

Currently, incoming call and outbound call are the available entry points.

Outputs

Action Handle

This handle contains all the actions for the rule.

Action Count

This integer is the count of actions held within the rule.

Exit Paths

Continue

This path continues on to the next step.

Abort

This step aborts the process.

Process Automation

Overview of Process Automation Tools

These tools are intended for use with the Interaction Process Automation modules in IC Business Manager and IC Server Manager. Click on a tool below for more information about that tool.

[Create Data Container](#)

[Get Data Element](#)

[Get Process Properties](#)

[Initiate Process](#)

[Send Process Automation Handler Results](#)

[Put Data Element](#)

[Query Server Info](#)

[Remove Data Container](#)

Create Data Container

Note: In prior versions, this tool was Create PA Data.

This Process Automation tool is for use with the Interaction Process Automation modules in IC Business Manager and IC Server Manager.

This Process Automation tool adds a new group of data elements that is available to all handlers and is accessible by name. Each data element in the group can contain a single value or a list of values.

Inputs

PA Data Name

The name by which the newly created group of data elements will be accessible.

Lifetime (minutes), 0 = infinite

If a non-zero value is specified, the group of data elements will exist until the time expires. Otherwise, it will exist until explicitly removed by the [Remove PA Data](#) tool.

Exit Paths

Success

This path is taken if the PA data is successfully created.

Failure

This step is taken if the operation fails.

PA Data Exists

This step is taken if the PA data name is already present in the list.

Get Data Element

Note: In prior versions, this tool was Get PA Data Element.

This Process Automation tool is for use with the Interaction Process Automation modules in IC Business Manager and IC Server Manager.

It gets the specified data element from the group of data elements.

Inputs

PA Data Name

The name of the group of data elements that contains the element to get.

Element Name

The name of the data element to get from the specified PA Data Name.

Outputs

Variable

The variable output can be one of these types: Integer, List of Integer, Numeric, List of Numeric, Boolean, List of Boolean, String, List of String, DateTime, List of DateTime.

Last Put Time

The Date/Time of the last successful update to any attribute in the attribute group.

Exit Paths

Success

If this step executes successfully, it takes the Success exit path.

Failure

If this step does not execute successfully, it takes the Failure exit path.

Unknown PA Data

This step is taken if the PA Data Name is not found.

Unknown Element

This step is taken if the Element Name is not found.

Incompatible Type

This step is taken if the output variable is not one of the valid types.

Complex Data Type Example

Starting in CIC 4.0, SU1, Interaction Process Automation supports complex data types. The following example describes how to use complex data types with this tool.

Suppose that your process has two complex data types: **Customer** and **Address**.

Address is defined as:

Address:	
City	(String)
State	(String)

Customer is defined as:

Customer:	
FirstName	(String)
LastName	(String)
HomeAddress	(Address)
AdditionalAddresses	(Address collection)

Suppose that your process includes a variable called **Customer1**, and **Customer1** has these values:

Customer1:	
FirstName:	John
LastName:	Doe
HomeAddress:	City: Indianapolis State: Indiana
AdditionalAddresses:	(an empty collection)

The following string is sent to Interaction Processor (IP) (as a string data type):

```
-->
<Customer1>
  <FirstName>John</FirstName>
  <LastName>Doe</LastName>
  <HomeAddress>
    <City>Indianapolis</City>
    <State>Indiana</State>
  </HomeAddress>
  <AdditionalAddresses />
</Customer1>
<!--
```

You can manipulate the string and send it back to the IPA process, and **Customer1** should update as intended. If you want to have two addresses in the **AdditionalAddresses** property, then see the following example:

```
-->
<Customer1>
  <FirstName>John</FirstName>
  <LastName>Doe</LastName>
  <HomeAddress>
    <City>Indianapolis</City>
    <State>Indiana</State>
  </HomeAddress>
  <AdditionalAddresses>
    <City>Chicago</City>
    <State>Illinois</State>
  </AdditionalAddresses>
  <AdditionalAddresses>
    <City>Detroit</City>
    <State>Michigan</State>
  </AdditionalAddresses>
</Customer1>
<!--
```

If the **AdditionalAddresses** property previously contained an item, that item is removed and the complex data type is updated to whatever is sent in the string. If the **AdditionalAddresses** element does not exist within the XML blob, then any items in that collection are removed.

If your process contains a Customer Collection variable, The Get PA Data Element tool binds the Customer Collection variable to a list of string in IP. The list and XML tools work the same way to edit strings that are returned from Interaction Process Automation.

You can use the [Put PA Data Element](#) tool to assign a value or list of values to the Customer Collection data element. If the input to the tool is a string, the input string is appended to the end of the Customer Collection that you're updating. Use the Put PA Data Element tool to assign a list of string value from IP to the Customer Collection data element in IPA.

Get Process Properties

This Process Automation tool is for use with Interaction Process Automation to get the properties for a process. The properties are then used by the [Initiate Process](#) tool.

Inputs

Process Name

The name of the process for which you need to get the properties.

Outputs

Process ID

The unique ID associated with the process.

Description

The process description.

Published Version

The version number associated with the published process.

Exit Paths

Success

If this step executes successfully, it takes the Success exit path.

Not Found

If the process name is not found, this step takes the Not Found exit path.

Failure

If this step does not execute successfully, it takes the Failure exit path.

Initiate Process

This Process Automation tool is for use with the Interaction Process Automation modules in IC Business Manager and IC Server Manager.

This Process Automation tool invokes a published process.

Inputs

Process ID

The unique ID associated with the process (obtained using the [Get Process Properties](#) tool).

Initiator Name

User name for the process' initiating user (required).

PA Data Name

The name of the group of data elements.

Timeout Seconds

The number of seconds the tool waits before timing out (defaults to 15 seconds if value is blank or 0).

Queue Process if Overloaded

If this property is set to 'false', the process will fail to launch if the process automation server is in an overload state. If the property is not explicitly set to 'false' and the process automation server is in an overload state, the process automation server will defer the launch request and retry the launch when the overload condition is resolved.

Outputs

Job Exec ID

A number to confirm the job execution. Information only, but can be used to cross-reference with reports, database, or Process Automation Server logs.

Flow Exec ID

For information only and can be used to cross-reference with reports, database, or Process Automation Server logs.

Numeric Process ID

For information only and can be used to cross-reference with Interaction Process Automation Monitor.

Exit Paths

Success

If this step executes successfully, it takes the Success exit path.

No Process

This step takes the No Process exit path if the process ID is invalid.

Timeout

This step takes the Timeout exit path if the PA Server does not respond to the initiate request.

No Server

This step takes the No Server exit path if the PA Server is not running.

Failure

If this step does not execute successfully, it takes the Failure exit path.

Send Process Automation Handler Results

Note: In prior versions, this tool was Process Automation send handler Results.

This Process Automation tool is for use with the Interaction Process Automation modules in IC Business Manager and IC Server Manager.

This Process Automation tool provides a way to send data and a result code for a handler started by the Process Automation Server. When the Run Handler action is used in a process, the [Process Automation Initiator](#) runs, gets the PA Data, and returns the data back to the Process Automation Server.

Inputs

Job ID

The Job ID from the Process Automation Initiator.

Sequence ID

The Sequence ID from the Process Automation Initiator.

Result Code

The result code to send back to the Process Automation Server.

PA Data Name

The name of the data element group to send back to the Process Automation Server.

Timeout Seconds

The number of seconds the tool waits before timing out (defaults to 15 seconds if value is blank or 0).

Exit Paths

Success

If this step executes successfully, it takes the Success exit path.

Failure

If this step does not execute successfully, it takes the Failure exit path

Put Data Element

Note: In prior versions, this tool was Put PA Data Element.

This Process Automation tool is for use with the Interaction Process Automation modules in IC Business Manager and IC Server Manager.

This Process Automation tool assigns a value or list of values to a data element. It is the reverse of the [Get PA Data Element](#) tool.

Inputs

PA Data Name

The name of the Process Automation data element group that contains the element for which the value will be assigned.

Element Name

The name of the data element for which the value will be assigned.

Value

The value or list of values associated with the data element. The value input can be one of these types: Integer, List of Integer, Numeric, List of Numeric, Boolean, List of Boolean, String, List of String, DateTime, List of DateTime.

Exit Paths

Success

If this step executes successfully, it takes the Success exit path.

Failure

If this step does not execute successfully, it takes the Failure exit path.

Unknown PA Data

This exit path is taken if the specified PA Data Name is not found.

Incompatible Type

This path is taken if the value specified is not one of the valid types.

Query Server Info

This Process Automation tool queries various load metrics on the server so that automated launchers can throttle launches.

Inputs

Timeout Seconds

The time (in seconds) to wait for a response.

Outputs

Init Status

An integer that indicates how far along the Process Automation Server is in its initialization at the time the tool is invoked. The value "4" indicates Active. All other values are currently reserved for future use.

This output is useful for checking that PAS is fully initialized before attempting to launch processes.

Load Status

An integer that is a function of CPU usage and memory usage and sums up how "busy" the PAS is.

The possible values are as follows:

Value	Indicates
0	Idle
1	Active - Low
2	Active - High
3	Busy
4	Overload
5	Unknown

Memory Percent

Memory usage as a percentage of a maximum value that Task Manager displays as Private Bytes.

The default maximum value is 1.4G, so while it would be rare, it is possible to see values greater than 100%.

Cpu Percent

CPU usage, calculated as a rolling average of ten samples spaced no closer than three seconds. Therefore, the values used to calculate the average span 30 seconds or more.

Thread Count

The total number of OS threads in use at a given time. Note that this is not the number of processes (see Active Processes below) and it's not the number of threads allocated, which you might see in PerfMon or Task Manager. Most of the time, this value should not be very high relative to the number of active processes. It might approach the number of threads allocated under very heavy loads.

You can compare this value to the Active Processes value to assess whether or not it is high. In some cases, it is normal to be high, but in general the design is such that PAS does not require very many threads to accommodate a large number of active processes as long as they are not all doing something at precisely the same instant.

Active Processes

The total number of processes that are running.

Protection Active

This is a boolean value that can be used to throttle inputs, if possible. If this value is true, it indicates that PAS has gone into self-protection mode to avoid running out of resources based on the current load. In this mode, PAS will reject launches until the pending backlog has been worked off and the values used to calculate the overload condition have fallen below the configured thresholds again.

Exit Paths

Success

This path is taken if the metrics are successfully retrieved.

Timeout

This path is taken if the metrics are not successfully retrieved before the timeout period has passed.

Failure

This path is taken if the operation fails.

Remove Data Container

Note: In prior versions, this tool was Remove PA Data.

This Process Automation tool is for use with the Interaction Process Automation modules in IC Business Manager and IC Server Manager.

This Process Automation tool removes a previously created Process Automation data element group. Note that if a Process Automation data element group is removed by any handler, it will not exist for any handler that tries to retrieve it after it has been removed.

Inputs

PA Data Name

The name of the Process Automation data element group created by the [Create PA Data](#) tool. The data element group must exist or the "Unknown PA Data" path will be taken.

Exit Paths

Success

This step is taken if the data element group is successfully removed.

Failure

This step is taken if the operation fails.

Unknown PA Data

This path is taken if the Process Automation data element group name was not found.

Reco

Introduction to Reco tools

Reco tools provide a handler developer the building blocks for speech applications. For more information about using Reco tools with the speech recognition subsystem, refer to the *CIC Speech Recognition Overview Technical Reference* in the PureConnect Documentation Library.

Reco Add Preloaded Grammar

This Reco tool adds a preloaded grammar. The grammar will be preloaded on all ASR servers of the specified engine, or all engines if no engine name is specified in the 'ASR Engine' parameter. Each preloaded grammar has a name as which it can be referenced in recognition operations through the `x-inin-reco:preloaded/name` URI.

Preloaded grammars are not associated with an interaction but are global to all interactions and are persistent. The primary purpose of preloaded grammars is to prime the ASR servers with large grammars, such as grammars of (large) company directories whose compilation may take several seconds or even minutes. Changing a pre-loaded grammar at runtime ensures that the new grammar is compiled and distributed to each ASR server before the old preloaded grammar is replaced. For example, consider a corporation with a large number of employees for which a dial-by-name feature is to be provided. Every night, say at 2am, the company directory is harvested and a dial-by-name grammar generated for all users. Compiling such a large grammar can take a long time and the first caller entering the system after the grammar change would experience an unacceptably long delay while the grammar is being compiled. Using preloaded grammars, the timer-scheduled handler would synthesize a new grammar file and invoke this tool. The grammar is then parsed, compiled, distributed to all ASR servers and the caches are primed. After the grammar has been successfully registered, the DS entry is modified to ensure the new URI is used to represent the preloaded grammar and the internal link of the preloaded grammar name to the actual grammar object is modified. The next interaction referencing the preloaded grammar will get the new grammar and there should be no noticeable delay in activating the new grammar for recognition.

It is possible to register a preloaded grammar of the same name for different engines or register one for all engines ('ASR Engine' parameter empty) and then register a different one for a specific engine. The engine specific grammar hides the general grammar for that particular engine. The appropriate preloaded grammar will then be selected based on the ASR engine associated with a session. This is useful, for example, if the grammars are rendered by a third-party tool and/or compiled using engine specific tools (e.g., command-line compilers), and then registered as compiled grammar files (e.g., Nuance NGO or SpeechWorks binary grammars). It is strongly recommended to match the semantics of the different grammars sharing the same name to prevent confusion. Thus, a preloaded grammar of a certain name should have the same semantics and slots for all engines.

In most cases, the file of the previous preloaded grammar is not used any longer (the timer task should not overwrite the old file, as a failure would lead to a corrupt grammar). This tool thus allows automatically deleting the replaced old grammar file.

Note 1: Refer to the description of the [Reco Register Grammar](#) tool for a list of the most commonly returned errors.

Note 2: Preloading grammars should be considered an expensive operation as the grammar is distributed to every ASR server, compiled, and loaded into the cache. Frequently changing preloaded grammars could thus have detrimental implications on the system performance. If possible, automatically changing pre-loaded grammars should be scheduled to be done in off-peak hours (e.g. early in the morning).

IMPORTANT: Once a file has been registered as preloaded grammar, it must not be modified directly. The recognition framework does not monitor the file for changes. Furthermore, if the system is stopped or terminates unexpectedly while the file is being modified, the grammar will be corrupt and lead to failures. Instead, when creating a new grammar to be used as preloaded grammar, create a new file and then use this tool to replace the current file.

Inputs

Name

Name of the preloaded grammar. This name is used to identify it in DS and in the `x-inin-reco:preloaded/name` URI.

If a preloaded grammar with this name already exists, it will be replaced.

The name must consist only of alphanumeric characters (a-z, A-Z, 0-9) or '_'. Names are case sensitive.

Grammar URI

URI of the grammar to add to register as pre-loaded grammar. The supported schemas and grammar formats are engine

dependent. If the URI includes a fragment, it will be ignored. Filenames will automatically be converted to a file: URI. Relative URIs are converted to absolute URIs based on the 'DefaultBaseURI' configuration parameter.

Grammar Mode

Mode of the referenced grammar. Currently, only "voice" grammars are supported as preloaded grammars.

Grammar Type

Optional. Media type (MIME type) of the grammar referenced by the URL. It is recommended to specify the MIME type for grammars other than engine agnostic grammars. The following are the media types of the engine-agnostic grammar formats:

application/srgs SRGS ABNF

application/srgs+xml SRGS GrXML

application/x-jsgf JSpeech

ASR Engine

Name of the ASR engine for which this preloaded grammar should be registered.

Delete replaced grammar if file

If this box is checked, and the URI of the grammar being replaced represents a file, and the file name is different from the new URI, the old file will be deleted after successfully registering the preloaded grammar.

Leave this box unchecked to only replace the preloaded grammar.

Register Asynchronously

If this box is checked, the grammar will be preloaded asynchronously. The tool will return before the grammar is compiled and sent to ASR servers. The advantage of this mode is that the handler isn't blocked for an extended period of time. The disadvantage is that compilation and other failures might not be known by the time the tool returns.

If this box is left unchecked, the tool will not exit until the grammar has been preloaded successfully.

Outputs

Registered URI

Fully qualified URI of the pre-loaded grammar.

Error Code

If the tool fails, this output parameter contains an error code in the form of a VoiceXML event. This will be an empty string if no error occurred.

Error Text

If the tool fails, this output parameter will contain a simple textual description of what went wrong.

Exit Paths

Success

A new preloaded grammar with the specified name has been added.

Failure

Some other error occurred.

Reco Analyze Error Code

This Reco tool simplifies analyzing the errors returned by the tools. The error code passed as argument is prefix-matched against a list of error codes. The error codes and the corresponding tool exits are defined through the tool property dialog.

The error codes are prefix-matched, whereas whole segments have to match. For example: Given the error code "error.badfetch.grammar.syntax". The pattern "error.badfetch" will match this error code, but "error.noresource", "error.badfetch.file", or "error.badfetch.grammar.syntax.sisr" will not match.

Inputs

Error Code

This output parameter contains the error code returned by the server. This identifies the error that will be analyzed.

Exit Paths

The Exit Paths are configured as part of the tool properties.

Reco Analyze Result

This Reco tool can be used to determine how to proceed with the result of a recognition. Usually, if there is a single result with a high confidence, it is immediately accepted and the dialog proceeds. If there is a single result with a lower confidence, it should not immediately be accepted but rather a confirmation should be obtained (e.g., "did you say John Doe?"). If there are multiple results that match the criteria, the handler has to disambiguate, for example by querying the user for the individual results (e.g., "There are multiple matches. Did you mean Mary Smith?" – "No" – "Mary Smythe?" – "Yes").

IMPORTANT: This tool only works correctly if the hypotheses in the recognition result passed as argument are in descending order of confidence. This will always be the case if the result data comes from one of the recognition tools.

Inputs

Recognition Result

XML DOM node of the ASR recognition result XML document (e.g. returned by [Reco Input](#)).

Acceptance Confidence

Minimum confidence score for results that are accepted without confirmation. Default: 0.85

Confirmation Confidence

Minimum confidence score for results that have to be confirmed. All results lower than this threshold will be ignored. Default: 0.2

Threshold Diffusion

This parameter is used to achieve a smart inclusion of results that might fall below the threshold, but that are statistically too similar to acceptable results to be excluded without confirmation. For example, consider a scenario where a recognition result has two hypotheses, one with a confidence of 0.87 and the other 0.83, and the Confirmation Confidence is 0.85. Only one of those two hypotheses is above the threshold of 0.85, however the one that is not is close enough to it that it would be reasonable to prompt the user before discarding it.

After all hypotheses that are above the acceptance or confirmation threshold have been found, the score of the hypothesis with the lowest score is multiplied by $[1.0 - \text{ThresholdDiffusion}]$. This value is then used as a new threshold and all hypotheses that are above this value are included too.

The default value of for this parameter is 0.05.

Outputs

Top Hypothesis

XML DOM Node of the `<hypothesis>` element with the highest score in its band. NULL node if no matching hypothesis found. This parameter will contain the top hypothesis even if there were multiple matches.

Hypotheses

XML node iterator to iterate over the hypotheses that match the criteria. The Top Hypothesis is the first one that will be returned by the [XML Get Next Node](#) tool.

Hypothesis Count

Number of hypotheses in the Hypotheses list.

Exit Paths

Accept Single

This path is taken if a single hypothesis is above the accept threshold.

Accept Multiple

This path is taken if multiple hypotheses are above the accept threshold.

Confirm Single

This path is taken if a single hypothesis is below the accept threshold but above the confirm threshold. This hypothesis should be confirmed with the caller.

Confirm Multiple

This path is taken if multiple hypotheses are below the accept threshold but above the confirm threshold. These hypotheses have to be confirmed.

Rejected

This path is taken if there are hypotheses in the recognition result, but they are all below the Confirm Confidence.

None

This path is taken if the recognition result is empty (no hypotheses) or the XML node is NULL.

Failure

This path is taken if an error occurred (e.g., not a recognition result).

Reco Basic Input

This Reco tool is used for simple input where a single, built-in grammar is sufficient (for example, to obtain a confirmation (yes/no) or other simple input where it would be too cumbersome to register grammars and use the [Reco Input](#) tool).

The Value Type represents the name of the built-in grammar to use. This corresponds to the Type attribute of a VoiceXML `<field>`

element. Thus, invoking this tool with "digits?length=4" as Value Type, corresponds to registering and using the following two built in grammars: "builtin:grammar/digits?length=4" and "builtin:dtmf/digits?length=4" (this assumes an Input Modes parameter of 3 [voice and DTMF]).

The grammar attribute in the result hypothesis corresponds to the value type prefixed with a "\$," without the query parameter component ("\$digits" in the above example).

Note: Please see Reco Input for additional information about the semantics of the input tools.

Inputs

Interaction

Identifier of the interaction.

Input Modes

An optional space-delimited list of input modes to use for this request. The input modes specified when the session was created are used by default.

Value Type

Value type of the input. This corresponds to the name and parameters of a built-in grammar.

DTMF Termination Keys

Defines the keys a user can press to terminate the input. The pound key (#) is set by default.

For example, when the IVR requests a PIN, the user can enter the PIN and then press # to proceed immediately, rather than waiting for the timeout.

DTMF Escape Keys

Defines the keys a user can press to escape a dialog. No escape key is set by default.

For example, if a user makes a mistake when entering their PIN and needs to start over, he or she can press the escape key to be prompted again for the PIN.

Confidence Level

The minimum confidence that the highest scoring hypothesis of the recognition result must have for a successful recognition. If no hypothesis is above this threshold, the tool will take the Nomatch exit path. This parameter corresponds to the confidencelevel property of VoiceXML.

This parameter must have a value between 0.0 and 1.0 (inclusive). The default value is 0.5.

Top N Answers

Maximum number of distinct answers to include in the recognition result. Default: 2

Note: This does not mean that there will be at most N hypotheses in the recognition result. Each answer may have multiple hypotheses with the same (or very similar) confidence. For example: Assume a dial-by-name grammar where the first names are optional (e.g. "[Adam] Smith | [John] Smith | [John] Smythe"). If the caller says "Smith", there are two possible hypotheses with the same confidence (the same confidence level), one for "Adam Smith" and one for "John Smith". Thus, these two hypotheses would count as one answer. The ASR engine may also provide a second answer with a lower confidence for "John Smythe". Increasing the value of the 'Top N Answers' parameter will cause the engine to search for more answers and will thus increase the computation required by the ASR engine.

Timing

Value

Maximum time to wait (in seconds) for speech or DTMF input. (That is, if no key is pressed or speech is detected within this time, the recognition aborts. The default timeout period is 5 seconds.)

Mode

Specifies the semantics of the timeout value.

Relative (0) Default. Timer starts when the plays complete (silence timeout).

Absolute (1) Timer starts immediately, regardless of how long the plays play.

FinalPlay (2) The timer starts when the last of the queued plays starts to play. Starts immediately if no plays are playing.

Interdigit

Maximum inter-digit delay for DTMF (in seconds). The default timeout period is 2.5 seconds.

Termination

Termination timeout (in seconds) for DTMF input when DTMF grammar must terminate. The default timeout period is 0 seconds.

Incomplete

Maximum time to wait (in seconds) after caller stops talking and grammar is not yet in an accepting state before timing out and taking the No Match exit path. The default timeout period is 1.5 seconds.

Complete

Maximum time to wait (in seconds) after a valid speech input has been provided before the input is accepted after the caller stops talking. The default timeout period is 0.5 seconds.

Max Speech

Maximum allowed duration of user speech before aborting. This prevents excessive background noise from blocking the tool indefinitely. The default timeout period is 20 seconds.

Tone Detector

Frequency (Hz) (Tone Detector 1)

First tone detection frequency in Hz. The default value is 1100. Set this parameter to zero to disable tone detection.

Max Deviation (Hz) (Tone Detector 1)

Maximum frequency deviation of tone 1 in Hz. The default value is 50.

Frequency (Hz) (Tone Detector 1)

Second tone detection frequency in Hz. The default value is zero. When set to zero, tone detection is disabled.

Max Deviation (Hz) (Tone Detector 2)

Maximum frequency deviation of tone 2 in Hz. The default value is 50.

ON Duration(s)

Time (in seconds) during which the tone must be on. The default value is 0.2.

ON Deviation(s)

Maximum deviation of tone duration. The default value is -0.2.

OFF Duration(s)

Time (in seconds) during which tone must be off. The default value is 0.0.

OFF Deviation(s)

Maximum deviation of tone off duration. The default value is -0.0

Interval Count

Number of tone on/off intervals required for match. The default value is 0.

Properties

Inline recognition properties. Properties specified here are active for the duration of the recognition. These properties can be used for advanced control of the recognition or to enable engine specific, custom extensions.

Outputs

XML Document

XML `<result>` element node of the recognition result data. If an error occurred, an empty XML node will be returned with error information attached. Use [XML Get Error Info](#) to obtain extended error information

Hypothesis Count

Number of recognition hypotheses in the recognition result.

Top "_value" Slot

Value of the "_value" slot of the hypothesis with the highest confidence. Empty string if top hypothesis does not have a _value slot.

Code (event)

This output parameter contains the error code if the tool failed in the form of a VoiceXML style event.

Text (message)

Text accompanying the error code.

Exit Paths

Success

This path is taken if a valid input matching one or more of the specified grammars is recognized. This may be DTMF or voice.

Escape

This path is taken if the DTMF Escape key is pressed.

Tone

This path is taken if a tone matching the specified rules is detected during the recognition.

No Input

This path is taken if "no input" was recognized during the specified timeout time. This corresponds to the noinput VoiceXML event.

No Match

This path is taken if the input provided but it did not match any of the active grammars. This corresponds to the nomatch VoiceXML event. Even if the tool returns through this exit, there may still be a recognition result.

Max Speech

This path is taken if the Max Speech Timeout was exceeded during speech input. This normally means that the background noise is too high and the handler should fall back to DTMF-only input. This exit corresponds to the maxspeechover VoiceXML event.

Failure

This path is taken if some other error occurred. Use [XML Get Error Info](#) on the returned recognition result node to obtain additional information.

Reco Bind Slot Values

This Reco tool binds the value of one or more slots of a hypothesis to string variables. It contains a list of slot-names to variable names. At runtime, the value of the named slot is bound to the corresponding string variable. The variables of slots that are not found are set to an empty string.

Parameters

Hypothesis

XML DOM element of a hypothesis.

Exit Paths

Success

This path is taken if a matching slot was found.

Failure

This path is taken if an error occurred.

Reco Create Company Directory Grammar

This Reco tool creates an SRGS ABNF grammar file containing the users in the company directory according to the criteria specified as parameters. The grammar rule created by the tool has the following structure:

```
LeadingFiller Names TrailingFiller
```

The names are rendered depending on the various check boxes of the parameters. Assuming all options are enabled, the following rule fragment is rendered for each user:

```
[Jane] Doe | Jane [Doe] | Doe Jane.
```

Thus, callers can either say "Jane," "Doe," "Jane Doe," or "Doe Jane." Possible alternate spellings specified in Interaction Administrator for that user are added as additional alternatives.

Note: If an alternate spelling ends with a language attachment, this tool treats the exclamation point and subsequent language identifier as a language attachment.

Enabling the 'Last Name Optional' or 'First Name Optional' parameters will allow broader recognition (that is, the callers do not have to say the full name). However, for large organizations this leads to lots of ambiguous matches, in particular for common names such as "Smith."

Note: All selected slots are always returned, even if the caller doesn't say the corresponding component. For example, if the first name is optional and the user just says the last name of a person, the "firstname" slot will nevertheless be filled with the first name of the directory entry.

IMPORTANT: Do not use this tool to overwrite a preloaded grammar or a grammar that might be in use. Instead, create a new grammar file with a unique name, (for example, by appending the current time and date). Rendering the company directory into the file may take quite some time. Overwriting a grammar file that may be used by the application could thus lead to a partially completed file being loaded and cause the grammar parser to fail. By creating a new file, and only referencing it once it is completed, prevents such problems. Use the [Reco Add Preloaded Grammar](#) tool to add/replace a preloaded grammar with the newly created company directory grammar.

Inputs

Grammar File

File name and path of the grammar file to be created. If no path is specified, the grammar will be written to the directory specified as 'DefaultGrammarBaseURI' Recognition configuration parameter. If the file name does not have an extension, ".gram" will be added.

Language

ISO language code of the grammar to be created (set as value of 'language' grammar header).

Workgroups

List of workgroups whose users are to be included in the grammar.

If undefined, all users will be included.

Leading Filler

ABNF Rule fragment rendered as filler before the actual name tokens. For example:

```
[dial | call | (transfer | connect) [me] [to | with]]
```

Note: The filler must be in the SRGS ABNF format and is not rendered as optional. Thus, specifying "dial" as filler will mean that the user *must* say "dial" followed by the name for the grammar to match.

If undefined, no leading filler is rendered.

Trailing Filler

ABNF Rule fragment rendered as filler after the actual name tokens. For example:

```
[please]
```

Note: The filler must be in the SRGS ABNF format and is not rendered as optional.

If undefined, no trailing filler is rendered.

Custom Headers

Custom header fields and rules to inject into the SRGS ABNF grammar. The following headers are created by the tool and cannot be injected: 'mode', 'language', 'root'.

First Name Optional

This checkbox specifies whether the grammar should be rendered with an optional first name. If left unchecked, the caller must say the first name of the person. If checked, the first name of the user is optional and callers may just say the last name.

Last Name Optional

This checkbox specifies whether the grammar should be rendered with an optional last name. If left unchecked, the caller must say the last name. If checked, the last name of the user is optional in the grammar and callers may just say the first name.

Recognize "Last Name, First Name" too

This checkbox specifies whether user may say "last name, first name" in addition to "first name, last name." If left unchecked, the grammar expects "first name, last name." If checked, the user may say either "first name, last name" or "last name, first name."

Include "user" slot

If checked, the recognition result will include a "user" slot containing the name (user id) of the user.

Include "extension" slot

If checked, the recognition result will include an "extension" slot containing the extension of the user.

Include "firstname" slot

If checked, the recognition result will include a "firstname" slot containing the first name of the user.

Include "lastname" slot

If checked, the recognition result will include a "lastname" slot containing the last name of the user.

Outputs

Grammar File URI

Full file URI of the created grammar file. Useful for passing to [Reco Register Grammar](#) or [Reco Add Preloaded Grammar](#).

Error Code

Returns code if tool exits through Failure.

Error Text

Message providing details about the error.

Exit Paths

Success

The grammar file has been created successfully.

Failure

Creation of grammar file failed. Consult the Error Code parameter for details.

Reco Create Company Directory Grammar 2

This Reco tool creates an SRGS ABNF grammar file containing the users in the company directory according to the criteria specified as parameters. This tool is similar to [Reco Create Company Directory Grammar](#), but can also generate grammars that include users' extensions when dialing. For example, it can be used to allow a user to call an IVR system and say "Dial extension 8157."

The grammar rule created by the tool has the following structure:

```
LeadingFiller NamesAndExtensions TrailingFiller
```

The names and extensions are rendered depending on the various checkboxes of the parameters. Assuming all options are enabled, the following rule fragment is rendered for each user:

```
[Jane] Doe | Jane [Doe] | Doe Jane | "Extension xxxx"
```

Thus, callers can either say "Jane," "Doe," "Jane Doe," "Doe Jane," or "extension xxxx." Possible alternate spellings specified in Interaction Administrator for that user are added as additional alternatives.

Enabling the **Last Name Optional** or **First Name Optional** parameters will allow broader recognition (i.e. the callers don't have to say the full name). However, for large organizations this will lead to lots of ambiguous matches, in particular for common names such as "Smith".

Enabling the **Recognize User Extension** parameter allows recognition based on the extension of a user. A caller can say "Extension" followed by the specific extension of the user.

Note: All selected slots are always returned, even if the caller doesn't say the corresponding component. For example, if the first name is optional and the user just says the last name of a person, the "firstname" slot will nevertheless be filled with the first name of the directory entry.

IMPORTANT: Do not use this tool to overwrite a preloaded grammar or a grammar that might be in use. Instead, create a new grammar file with a unique name, (for example, by appending the current time and date). Rendering the company directory into the file may take quite some time. Overwriting a grammar file that may be used by the application could thus lead to a partially completed file being loaded and cause the grammar parser to fail. By creating a new file and only referencing it once it is completed prevents such problems. Use the [Reco Add Preloaded Grammar](#) tool to add/replace a preloaded grammar with the newly created company directory grammar.

Inputs

Grammar File

File name and path of the grammar file to be created. If no path is specified, the grammar will be written to the directory specified as 'DefaultGrammarBaseURI' Recognition configuration parameter. If the file name does not have an extension, ".gram" will be added.

Language

ISO language code of the grammar to be created (set as value of 'language' grammar header).

Workgroups

List of workgroups whose users are to be included in the grammar.

If undefined, all users will be included.

Leading Filler

ABNF Rule fragment rendered as filler before the actual name tokens. For example:

```
[dial | call | (transfer | connect) [me] [to | with]]
```

Note: The filler must be in the SRGS ABNF format and is not rendered as optional. Thus, specifying "dial" as filler will mean that the user *must* say "dial" followed by the name for the grammar to match.

If undefined, no leading filler is rendered.

Trailing Filler

ABNF Rule fragment rendered as filler after the actual name tokens. For example:

```
[please]
```

Note: The filler must be in the SRGS ABNF format and is not rendered as optional.

If undefined, no trailing filler is rendered.

Custom Headers

Custom header fields and rules to inject into the SRGS ABNF grammar. The following headers are created by the tool and cannot be injected: 'mode', 'language', 'root'.

First Name Optional

This checkbox specifies whether the grammar should be rendered with an optional first name. If left unchecked, the caller must say the first name of the person. If checked, the first name of the user is optional and callers may just say the last name.

Last Name Optional

This checkbox specifies whether the grammar should be rendered with an optional last name. If left unchecked, the caller must say the last name. If checked, the last name of the user is optional in the grammar and callers may just say the first name.

Recognize "Last Name, First Name" too

This checkbox specifies whether user may say "last name, first name" in addition to "first name, last name." If left unchecked, the grammar expects "first name, last name." If checked, the user may say either "first name, last name" or "last name, first name."

Recognize User Extension

If this checkbox is selected, users can request a particular extension by saying, for example, "Dial extension 8157."

User Extension Prefix

This parameter specifies the grammar (one or more words) that callers can use before the extension if the **Recognize User Extension** parameter is selected. By default, the prefix is "extension."

User Extension Suffix

This parameter specifies the grammar (one or more words) that callers can use after the extension if the **Recognize User Extension** parameter is selected. There is no default suffix.

Include "user" slot

If checked, the recognition result will include a "user" slot containing the name (user id) of the user.

Include "extension" slot

If checked, the recognition result will include an "extension" slot containing the extension of the user.

Include "firstname" slot

If checked, the recognition result will include a "firstname" slot containing the first name of the user.

Include "lastname" slot

If checked, the recognition result will include a "lastname" slot containing the last name of the user.

Outputs

Grammar File URI

Full file URI of the created grammar file. Useful for passing to [Reco Register Grammar](#) or [Reco Add Preloaded Grammar](#).

Error Code

Returns code if tool exits through Failure.

Error Text

Message providing details about the error.

Exit Paths

Success

The grammar file has been created successfully.

Failure

Creation of grammar file failed. Consult the Error Code parameter for details.

Reco Create Simulated Interaction

This Reco tool is used for testing and debugging. It creates a simulated interaction in the RecoTSSimulator process. The RecoTSSimulator runs simulation scripts containing a list of states the interaction may be in, as well as responses to stimuli. Each of these state lists is named (its class) to support multiple state lists in the same simulation scripts.

The simulation script may either be supplied as a Simulation Script argument or the default script loaded by the RecoTSSimulator process at startup (defined on command line).

The interaction returned by this tool can be treated like a normal call. Prompts can be queued and recognition operations performed.

This tool also returns the <CustomData> child element if the <Interaction> element in the simulation script has one. Custom data is very useful to embed information in the simulation script to control automatic tests.

This tool fails if the RecoTSSimulator is not running.

IMPORTANT: The RecoTSSimulator is a debugging tool and **MUST NOT** be run together with the regular TS Server. Running the RecoTSSimulator on a production server will lead to loss of dial tone!

Inputs

Simulation Script

The RecoTSSimulator element of the simulator script to be executed. If this parameter is undefined, the default simulation script of the TS simulator will be used.

Class

The name of the simulation class in the Reco TS Simulator script for which a simulated interaction is to be created. If this parameter is undefined, the first <Interaction> element in the simulation script will be used.

Language

Language identifier of the interaction to be created. The default language is United States English (en-US).

Outputs

Interaction

Identifier of the simulated interaction.

Custom Data

<CustomData> child element of the <Interaction> element in the simulation script. This node will be NULL if the <Interaction> element does not have a <CustomData> child element.

Error Code

This output parameter contains the error code if the tool failed.

Error Text

Text accompanying the error code.

Exit Paths

Success

This path is taken if the simulated interaction was successfully created.

Failure

This path is taken if an error occurred. The most probable cause for failure is if the RecoTSSimulator is not running.

Reco Custom Operation

This Reco tool allows the engine integration modules to expose custom operations offered by the ASR engine they implement. For example, it can be used to manage the voiceprint database for speaker identification support.

Inputs

Interaction

Identifier of the interaction.

Operation Name

Name of the custom operation to execute.

Parameter Names

Optional string list containing the names of the parameters. If no names are specified, the parameters are considered unnamed.

Parameter Values

Optional string list of parameter values passed to the engine integration module.

Outputs

Return Value

The value returned from the function.

Error Code

This output parameter contains the error code returned by the server.

Error Text

Text accompanying the error code.

Exit Paths

Success

This path is taken if the custom operation has been executed successfully.

Failure

This path is taken if the operation failed. For more details, the handler may analyze the event returned in the Error Code output.

Reco Disconnect Simulated Interaction

This Reco tool disconnects a simulated interaction created with the [Reco Create Simulated Interaction](#) tool.

Inputs

Interaction

Identifier of the simulated interaction.

Exit Paths

Next

This tool always takes the Next exit path.

Reco Filter Result

This Reco tool creates a new result XML document that contains `<hypothesis>` elements from the Argument parameter based on a Pattern and some match criteria. There are two options:

1. Include those hypotheses that match; or
2. Those that don't match (depending on the Exclude Matching Hypotheses parameter).

This tool is very useful to cull hypotheses from a recognition result that were already seen before. The `<hypothesis>` elements in the output result document are sorted by confidence.

Inputs

Argument

XML DOM node of a `<result>` element. The argument document is not modified.

Pattern

XML DOM node of a `<result>` or `<hypothesis>` element. The argument document is not modified.

Compare Mode

Specifies whether to compare the values of the Mode attributes. Leave this box checked if you only want the hypotheses to match if the values for the mode attributes are equal. Clear this box if you don't want the modes to be compared.

Compare Confidence

Specifies whether to compare the values of the Confidence attributes. Leave this box unchecked if you do not want the two hypotheses confidences to be considered. Check this box if you only want to match hypotheses that have equal confidences.

Compare Grammar ID

Specifies whether to compare the values of the Grammar attributes. Leave this box checked if you want the two hypotheses to have matching Grammar ID values. Clear this box if you do not want the Grammar IDs to be considered.

Compare Utterance

Specifies whether to compare the values of the "utterance" attributes. Leave this box unchecked if you do not want the values of the `<utterance>` elements to be compared. Check this box if you only want to match hypotheses with matching `<utterance>` elements.

Compare Slots

Specifies whether to compare the number, names, and values of the slots of the hypotheses. Leave this box checked if you only want to match hypotheses with matching slots. Clear this box if you do not want the slots to be compared.

Slot Names

Space-separated list of names of the slots to compare. If not specified, compares all slots.

Note: Use the "dot" notation to reference individual nested slots.

Exclude Matching Hypotheses

Specifies whether to negate the match. Leave this box checked if you want to include only those hypotheses from the Argument in the filtered result that do not match any hypotheses in the Pattern document. This corresponds to a set difference.

Clear this box to include hypotheses from Argument in the filtered result that match with one or more hypotheses from the Pattern document. This roughly corresponds to a set intersection.

Outputs

Result

XML DOM node of the `<result>` element of a new document that contains the filtered recognition result.

Hypothesis Count

Number of `<hypothesis>` elements in the result.

Exit Paths

Success

This path is taken if there is a valid result with at least one hypothesis.

Empty Result

This path is taken if the Recognition Result document has become empty.

Failure

This path is taken if some error occurred, most likely an invalid `<result>`.

Reco Get Hypothesis

This Reco tool retrieves a hypothesis from the recognition result based on the specified parameters. It returns the first hypothesis for which all specified parameters match. The Index can be used to easily iterate over the hypotheses.

Inputs

Recognition Result

XML DOM node of the ASR recognition result XML document.

Index

Zero-based index of the hypothesis to retrieve. The default value is the first hypothesis that matches other parameters.

Mode

Mode to match ("dtmf" or "voice"). "Any" mode will be used by default.

Min. Confidence

Minimum confidence score the hypothesis must have. Default: 0.0

Grammar ID

ID of the grammar whose hypothesis to return. Default: any grammar

Outputs

Hypothesis Element

XML DOM Node of the `<hypothesis>` element that matches the specified parameters. NULL node if no matching hypothesis found.

Hypothesis Index

Zero-based index of the hypothesis element as child element of the recognition result. A value of `-1` will be returned if there are no hypotheses.

Hypothesis Mode

Mode of the hypothesis ("dtmf" or "voice"). An empty string will be returned if there are no hypotheses.

Hypothesis Confidence

Confidence score of this hypothesis (0.0 to 1.0).

Hypothesis Grammar ID

ID of the grammar that accepted this hypothesis.

Exit Paths

Success

This path is taken if a matching hypothesis was found.

Not Found

This path is taken if no hypothesis matching the specified parameters was found.

Failure

This path is taken if an error occurred.

Reco Get Next Hypothesis

This Reco tool retrieves the next hypotheses from a list of hypotheses.

Inputs

Hypothesis Iterator

Iterator to collection of `<hypothesis>` elements.

Outputs

Hypothesis Iterator

Iterator pointing to next hypothesis in the list.

Element

XML DOM Node of the `<hypothesis>` element.

Mode

Mode of the hypothesis ("dtmf" or "voice"). This will be an empty string if there is no hypothesis.

Confidence

Confidence score of this hypothesis (0.0 to 1.0).

Grammar ID

ID of the grammar that accepted this hypothesis.

Exit Paths

Success

This path is taken if there was a hypothesis in the list.

End

This path is taken if the iterator points to end of the list (or list is empty).

Failure

This path is taken if an error occurred.

Reco Get Property

This Reco tool retrieves the value of a recognition property. These properties are managed by the recognition subsystem and used to control engine specific or extended functions.

Inputs

Interaction

Identifier of the interaction.

Property Name

Name of the property to retrieve. This string is case-sensitive.

Outputs

Property Value

Current value of the property.

Error Code

This output parameter contains the error code if the tool failed.

Error Text

Text accompanying the error code.

Exit Paths

Success

This path is taken if the value of the property was successfully retrieved.

Failure

This path is taken if an error occurred.

Reco Get Registered Grammars

This Reco tool retrieves a list of the grammars registered on the session of the specified interaction.

Note: If the interaction does not yet have a session, an empty string will be returned and the tool will take the None exit path.

Inputs

Interaction

Identifier of the interaction.

Outputs

Grammar Ids

Space-separated list of IDs of grammars registered with the current interaction.

Grammar Count

Number of registered grammars.

Error Code

If the registration fails, this output parameter contains an error code in the form of a VoiceXML event. This will be an empty string if no error occurred.

Error Text

If the tool fails, this output parameter will contain a simple textual description of what went wrong (e.g., the reason for compilation error).

Exit Paths

Success

This path is taken if the list of active grammars was successfully retrieved.

None

This path is taken if the ASR session of this interaction does not have any grammars.

Failure

This path is taken if an error occurred.

Reco Get Slot Value

This Reco tool retrieves a slot element and its value based on the specified criteria. Three different elements may be passed as arguments:

- `<result>` element: The slot is searched in the first hypothesis of the recognition result. The semantics are exactly the same as if the first hypothesis child element were passed.
- `<hypothesis>` element: The slot is searched in the hypothesis element.
- `<slot>` element: The child slot elements of this slot are searched.

Inputs

Hypothesis Or Slot

XML DOM element of a hypothesis, slot, or recognition result.

Index

Zero based index of the slot (child of hypothesis or parent slot). Defaults to the first slot that matches the name

Name

Name of the slot (case-sensitive). To reference a specific child slot in a nested slot, use a period as qualifier. For example. "foo.bar" accesses the slot "bar" which is a child slot nested in the slot "foo." The name is ignored by default, i.e., slot at index or confidence.

Min. Confidence

Minimum confidence the slot has to have for a match. No confidence attribute assumes 1.0. Default: 0.0

Outputs

Slot Element

XML DOM element of the matching slot. NULL Node if no matching slot found.

Slot Value

Value of the slot. If the slot has nested slots, this value is the recursive concatenation of the values of all child slots.

Slot Index

Zero-based index of the slot.

Slot Name

Name of the slot. An empty string will be returned if the slot is unnamed.

Slot Confidence

Confidence score of this hypothesis (0.0 to 1.0).

Exit Paths

Success

This path is taken if a matching slot was found.

Not Found

This path is taken if no slot matching the specified parameters was found.

Failure

This path is taken if an error occurred.

Reco Has Feature

This Reco tool checks whether the recognition session of this interaction has certain features. This allows adapting the handler logic to the capabilities of the ASR engine (or recognition capabilities in general) of a session.

Features can only be queried if the interaction has a recognition session. This tool does not automatically create a session.

Inputs

Interaction

Identifier of the interaction.

Features

Space-separated list of features. The engine must support all listed features for tool to take the Yes exit path.

Outputs

Unsupported Features

Space separated list of features that are not supported. This will be an empty string if all features are supported.

Exit Paths

YES

This path is taken if all of the specified features are supported by the current ASR engine of the specified session.

NO

This path is taken if one or more of the specified features are not supported.

No Session

This path is taken if the interaction does not have a session.

Failure

This path is taken if an error occurred.

Reco Initialize

This Reco tool explicitly initializes an ASR session for an interaction. It allows specifying the engine to use. If no engine is given, the recognition subsystem will pick an engine based on the language identifier of the interaction, the requested features, and the configuration order of the engines.

The Reuse Compatible Session checkbox controls whether an existing ASR session must be re-initialized or can simply be reused. If all parameters are left unspecified and Reuse Compatible Session is checked, an ASR session with default parameters will be created if the interaction does not have a session. If the current session is compatible, the session is used as is. If the checkbox is unchecked, a new ASR session will be created irrespectively of whether the interaction already has an active session, even if the specified parameters match the exiting session.

Common error codes returned by this tool:

Error code	Description
error.com.inin.reco.asr.engine	Specified ASR engine is unknown or not installed
error.unsupported.language	Engine doesn't support the specified language
error.com.inin.interaction.type	The specified interaction type is not supported by the recognition subsystem.
error.com.inin.inputmodes	Not all of the specified 'Input Modes' are supported ('No Fallback' = true) or none of the specified 'Input Modes' are supported ('No Fallback' = false).
error.noresource	Unable to initialize the recognition session because there are too many sessions already active and the number of available ports is limited (e.g. licenses or total load of all ASR servers exceeded limit).
error.com.inin.reco.feature	The requested capabilities are not supported by the specified engine (or any engine that may be selected based on the language etc.).
error.com.inin.reco.session.tied	The session configuration cannot be changed, as the ASR engine of the currently active session doesn't support this (once the engine is initialized, it has to stay with the call). The engine may still support changing the recognition language.
error.com.inin.ownership	The invoking handler is not the owner of the interaction or it lost the ownership.

Note: Irrespective of whether a new session is created or an old session reused, invoking this tool will always un-register all grammars and reset the session.

IMPORTANT: If the Language attribute of an interaction is changed after this tool has been invoked, the language of the ASR session will not change until the session is re-initialized by calling this tool.

Inputs

Interaction

Identifier of the interaction (call)

ASR Engine Name

Optional. Name of the ASR engine configuration to use. If not defined, consider any engine that matches the Language ID argument or the Language interaction attribute and the required features.

Server Groups

Note: This input is not used and is reserved for future use.

Language ID

Optional. ISO language ID of the language to be recognized. If this parameter is not defined, then by default the value of "Language" interaction attribute will be used.

Note: Some engines support multiple languages and/or use the language specified in the grammar. This parameter may thus just be used to ensure that the engine supports the language (but is still may allow using grammars of other languages).

Required Features

Optional. String containing a space separated list of features that the ASR engine must support. Default is an empty string.

Input Modes

Optional. Space-separated list of input modes to use for this session. If this parameter is not defined, the value of the the 'Eic_RecoInputModes' interaction attribute is used by default.

Max. Wait for Resource

If allocating an ASR port fails because all of them are in use, the Reco subsystem may wait up to the number of seconds specified by this parameter for a port to become available. If no port is available within this time, the tool fails with 'Resource Limit'. The default value for this parameter is 0.0s

Reuse Compatible Session

This checkbox specifies behavior if the interaction already has an associated ASR session. This box is checked by default, and so the session won't be re-initialized and the engine won't be unloaded if this interaction already has an existing ASR session that matches the specified parameters. All grammars will be deactivated, however.

Clear this box if you want the session to always reset and the engine to reload.

Lazy ASR Port Allocation

This checkbox specifies whether a physical ASR port should only be allocated the first time an ASR operation is actually performed. This box is unchecked by default.

Check this box if you want the tool to decide which engine to use, but not actually allocate the port. If this box is unchecked and all ports are in use, the tool will take the Failure exit path.

No 'Input Modes' Fallback

This checkbox specifies whether the specified input modes must be supported or an automatic fallback is OK. This only applies if 'Input Modes' argument is defined (does not apply to default). This box is checked by default and the tool will take the Failure exit path if any input modes specified as "Input Modes" arguments cannot be used.

Clear this box if you do not want this tool to fail if only some of the input modes specified as "Input Modes" are unavailable. If *all* such modes are unavailable, the tool will still take the Failure exit path.

Properties

Inline properties (name/value pairs) to apply to session. These properties can be used to control the engine selection, pass engine specific data to the engine integration, or set default values for certain parameters.

Outputs

Error Code

This output parameter contains the error code if the tool failed in the form of a VoiceXML style event.

Error Text

Text accompanying the error code.

Exit Paths

Success

This path is taken if the recognition session was successfully initialized.

Failure

This path is taken if an error has occurred. The Error Code and Error Text parameters contain details about the failure.

Reco Input

This Reco tool sends a request to the input recognition subsystem to accept user input either by speech or DTMF. The recognition will be performed using the grammars specified by their GrammarIDs. The list may include voice and or DTMF grammars and the recognition will use the corresponding input methods. Thus, if no DTMF grammar is specified, only voice input is possible and vice-versa. All grammars whose IDs are specified in the Grammars argument must have been registered previously through the [Reco Register Grammar](#), [Reco Register Grammar String](#), or [Reco Register Inline Grammar](#) tools.

In addition to GrammarIDs of explicitly registered grammars, grammars can also be referenced through "inline" references. Inline references have the following format:

```
$< GrammarURI > [~< GrammarType >] [^<GrammarID>]
```

Thus, a URI grammar maybe referenced without having to previously register it. This is probably most useful for built-in grammars.

Note 1: It is not necessary to call [Reco Initialize](#) before invoking this tool, as an ASR engine will automatically be chosen if none is already active (see Reco Initialize for details on how an engine is chosen if no other parameters are specified).

Note 2: If Input Modes other than the default are specified and one of the input modes is not supported by the current session, it is simply ignored. Thus, if Input Modes is "3" (Voice & DTMF), yet the session does not support ASR, only DTMF input is solicited. If none of the grammar IDs represents a DTMF grammar, the tool would fail with a Grammar Error. If the Input Modes parameter is "2" (Voice) and only DTMF is supported, the tool fails with a Failure error and the Error Code output parameter contains "error.com.inin.mode". It is thus recommended to use the default for Input Modes, unless it is used for a fallback to DTMF or to explicitly request a certain input mode.

Inputs

Interaction

Identifier of the interaction.

Input Modes

Optional. Space separated list of input modes to use for this request. Default: Input modes specified when the session was created.

Grammars

Space separated list of Grammar IDs of the grammars that are to be used to use to accept speech and DTMF input during this recognition step. In addition to GrammarIDs of previously registered grammars, explicit references may be specified too.

DTMF Termination Keys

DTMF termination key(s). Default: "#"

DTMF Escape Keys

DTMF escape key(s). Default: ""

Confidence Level

Minimum confidence the highest scoring hypothesis of the recognition result must have for a successful recognition. If no hypothesis above this threshold, the tool returns through the Nomatch exit. This parameter corresponds to the "confidencelevel" property of VoiceXML.

Range: 0.0 ... 1.0. Default: 0.5.

Top N Answers

Maximum number of distinct answers to include in the recognition result. Default: 2.

NOTE: This does not mean that there will be at most N hypotheses in the recognition result. Each answer may have multiple hypotheses with the same (or very similar) confidence. For example: Assume a dial-by-name grammar where the first names are optional (e.g. "[Adam] Smith | [John] Smith | [John] Smythe"). If the caller says "Smith," there are two possible hypotheses with the same confidence (the same confidence level), one for "Adam Smith" and one for "John Smith." Thus, these two hypotheses would count as one answer. The ASR engine may also provide a second answer with a lower confidence for "John Smythe." Increasing the value of the Top N Answers parameter will cause the engine to search for more answers and will thus increase the computation required by the ASR engine.

Timeout

Value

Maximum time to wait (in seconds) for speech or DTMF input (i.e. if no key is pressed or speech is detected within this time, the recognition aborts). Default: 5s

Mode

Specifies the semantics of the Timeout value.

Relative (0): Default. Timer starts when the plays complete (silence timeout).

Absolute (1): Timer starts immediately, irrespective of how long the plays play.

FinalPlay (2): The timer starts when the last of the queued plays starts to play. Starts immediately no plays are playing.

Interdigit Timeout

Maximum inter-digit delay for DTMF (in seconds). Default: 2.5s

Termination Timeout

Termination timeout (in seconds) for DTMF input when DTMF grammar must terminate. Default: 0s

Incomplete Timeout

Maximum time to wait (in seconds) after caller stops talking and grammar is not yet in an accepting state before timing out and returning through No Match. Default: 1.5s.

Complete Timeout

Maximum time to wait (in seconds) after a valid speech input has been provided before the input is accepted after the caller stops talking. Default: 0.5s

Max Speech Timeout

Maximum allowed duration of user speech before aborting. This prevents excessive background noise from blocking the tool indefinitely. Default: 20s

Tone Detector

Tone Detector 1 Frequency

First tone detection frequency in Hz. Set to 0 to disable tone detection. Default: 1100

Tone 1 Max Deviation

Maximum frequency deviation of tone 1 in Hz. Default: 50

Tone Detector 2 Frequency

Second tone detection frequency in Hz. Set to 0 to disable tone detection. Default: 0

Tone 2 Max Deviation

Maximum frequency deviation of tone 2 in Hz. Default: 50

ON Duration(s)

Time (in seconds) during which tone must be on. Default: 0.2

ON Deviation(s)

Maximum deviation of tone duration. Default: -0.2

OFF Duration(s)

Time (in seconds) during which tone must be off. Default: 0.0

OFF Deviation

Maximum deviation of tone off duration.
Default: -0.0

Interval Count

Number of tone on/off intervals required for match. Default: 0

Properties

Inline recognition properties. Properties specified here are active for the duration of the recognition. These properties can be used for advanced control of the recognition or to enable engine specific custom extensions.

Outputs

XML Document

`<result>` element node of the recognition result data.

If an error occurred, an empty `<result>` node will be returned with error information attached. Use [XML Get Error](#) to obtain extended error information.

Hypothesis Count

Number of recognition hypotheses in the recognition result.

Code (event)

This output parameter contains the error code if the tool failed in the form of a VoiceXML style event.

Text (message)

Text accompanying the error code.

Exit:**Success**

A valid input matching one or more of the specified grammars was recognized. May be DTMF or voice.

Escape

The DTMF "Escape" key was pressed.

Tone

A tone matching the specified rules was detected during the recognition.

No Input

No input was recognized during the specified timeout time. This corresponds to the "noinput" VoiceXML event.

No Match

Input was provided, but it did not match any of the active grammars. This corresponds to the "nomatch" VoiceXML event. Even if the tool returns through this exit, there may still be a recognition result.

Max Speech

The "Max speech timeout" was exceeded during speech input. This probably means that the background noise is too high and the handler should probably fall back to DTMF-only input. This exit corresponds to the "maxspeechevent" VoiceXML event.

Failure

Some other error occurred. Use XML Get Error on the returned recognition result node to obtain additional information.

Reco Log Event

This Reco tool gives speech applications the ability to write events into the ASR engine log. These events, for example, allow marking segments of the call that other utilities, such as OpenSpeech Insight, can report on.

Inputs

Interaction

A variable name representing the interaction and interaction type to log.

Event Name

Name of the event to be logged in the ASR engine log.

Event Value

The value of the event to be logged.

Outputs

Error Code

This output parameter contains the error code if the tool exits through Failure.

Error Text

A message providing details about the error code.

Exit Paths

Success

This path is taken if the event was logged successfully.

Failure

The logging operation failed. The details are provided in the Error Code and Error Text.

Reco Merge Results

This Reco tool creates a new result XML document that contains a union of two recognition results. The arguments may be `<result>` or `<hypothesis>` elements. The XML documents of the arguments are not modified (thus, a copy is made of the elements).

The merged result is created as follows:

1. Take the first hypothesis from argument 1 and add to result.
2. Compare each hypothesis in argument 1 and argument 2 against the hypotheses in the result and if it does not match any hypothesis in the result.
3. If a hypothesis matches one that is already in the result, it replaces the existing one if the confidence is higher (thus, the matching hypothesis with the highest confidence is kept).

Thus, merging the hypotheses can also be used to prune the set of results. For example, if only the Compare Mode checkbox is

checked and all others are unchecked, the merged result will contain at most one hypothesis for each mode, namely the one with the highest confidence.

Inputs

Result or Hypothesis 1

XML DOM node of a `<result>` or `<hypothesis>` element. The argument document is not modified.

Result or Hypothesis 2

XML DOM node of a `<result>` or `<hypothesis>` element. The argument document is not modified.

Compare Mode

Specifies whether to compare the values of the Mode attributes. Leave this box checked if you only want the hypotheses to match if the values for the mode attributes are equal. Clear this box if you don't want the modes to be compared.

Compare Confidence

Specifies whether to compare the values of the Confidence attributes. Leave this box unchecked if you do not want the two hypotheses confidences to be considered. Check this box if you only want to match hypotheses that have equal confidences.

Compare Grammar ID

Specifies whether to compare the values of the Grammar attributes. Leave this box checked if you want the two hypotheses to have matching Grammar ID values. Clear this box if you do not want the Grammar IDs to be considered.

Compare Utterance

Specifies whether to compare the values of the "utterance" attributes. Leave this box unchecked if you do not want the values of the `<utterance>` elements to be compared. Check this box if you only want to match hypotheses with matching `<utterance>` elements.

Compare Slots

Specifies whether to compare the number, names and values of the slots of the hypotheses. Leave this box checked if you only want to match hypotheses with matching slots. Clear this box if you do not want the slots to be compared.

Slot Names

Space delimited list of names of the slots to compare. If not specified, compare all slots. This is ignored when Compare Slots is not checked.

Note: Use the "dot" notation to reference individual nested slots.

Outputs

Result

XML DOM node of the `<result>` element of a new document that contains the union of the hypotheses.

Hypothesis Count

Number of `<hypothesis>` elements in the result.

Exit Paths

Success

This path is taken if the results are valid.

Empty Result

This path is taken if both arguments are empty result sets.

Failure

This path is take if an error occurred. Most likely an invalid `<result>` or `<hypothesis>` element specified as an argument.

Reco Query Input Modes

This Reco tool retrieves the `Eic_RecoInputModes` interaction attribute, checks it against the specified mask, and takes the Enabled or Disabled exit depending on the test. It takes the Enabled exit if all bits of Mode Check Mask in `Eic_RecoInputModes` are set. Thus, the following expression must evaluate to true: `(Mask AND Eic_RecoInputModes) == Mask`.

The tool always returns as Enabled if the mask is undefined, an empty string or "0".

The output parameter Input Modes always returns the value of the `Eic_RecoInputModes` interaction attribute, irrespective of the mask value. The input modes are returned as space-separated list of names.

Note: A Handler should not access the `Eic_RecoInputModes` attribute directly, as it might not be initialized.

Inputs

Interaction

Identifier of the interaction.

Mode Check Mask

Mask of modes to check against the currently set modes. By default, this parameter is set to just return the value, and thus the tool will always take Enabled exit path if left unchanged.

Outputs

Input Modes

Value of the `Eic_RecoInputModes` property as a space-separated list of names.

Note: This value is not affected by the mask.

Exit Paths

Enabled

This path is taken if the input modes specified as arguments are enabled.

Disabled

This path is taken if one or more of the input modes specified as arguments are not enabled.

Failure

This path is taken if an error occurred.

Reco Query Simulator Script State

This Reco tool queries the identifier of the current state in the simulation script. It also returns the <CustomData> child element if the <State> element has one. Custom data is very useful to control the test or embed expected results in the simulation script.

Inputs

Interaction

Identifier of the simulated interaction.

Outputs

Current State

Identifier of the current state in the simulation script ('id' attribute of the <State> element). This will be an empty string if the <State> element doesn't have an 'id' attribute.

Custom Data

<CustomData> child element of the current <State> element in the simulation script. This will be a NULL node if the <State> element does not have a <CustomData> child element.

Exit Paths

Success

This path is taken if the simulated interaction successfully queried.

Failure

This path is taken if there is an error querying the simulated interaction. This usually means that the interaction has either already been disconnected or it's not a simulated interaction.

Reco Register Grammar

This Reco tool registers a grammar specified by URL with the recognition subsystem. The grammar can be referenced in subsequent tool steps by its identifier. Once a grammar is registered, it can be referenced through its ID on the same session until it is explicitly unregistered with [Reco Unregister Grammar](#). Closing the session or re-initializing it with [Reco Initialize](#) will also clear all registered grammars.

The same grammar may be registered with multiple IDs, as the IDs simply represent a "moniker" for the grammar. The behavior of registering a different grammar with an ID that already exists depends on the Override Duplicate ID parameter.

In general, it is recommended to explicitly specify a GrammarID and leave the Override Duplicate ID checkbox checked. This permits optimizing the grammar registration if the exact same grammar is registered multiple times, for example if a subroutine handler is invoked multiple times during a call. Choosing a consistent naming scheme for GrammarIDs does not only simplify handler debugging and maintenance, but also optimizes grammar registration and cache performance. For example: For grammars specific to a handler, use the handler name as prefix to the GrammarID, such as "MyHandler_Gram1"; for common or built-in grammars, use a grammar ID that will be used throughout all (subroutine) handlers for the same grammar.

Built-in grammars have to be registered to create a reference (GrammarID) for it. The resulting Grammar ID will constitute an alias for the built-in grammar with its parameters. Registration is *not* necessary if the [Reco Basic Input](#) tool is used or an inline reference is used in the [Reco Input](#) tool. Explicitly registering grammars that are used multiple times is more efficient, though.

When the grammar is registered with Override Duplicate IDs and a grammar with the specified ID already exists, the existing grammar is replaced with the new grammar. Thus, the GrammarID now refers to the new grammar.

Grammars do not have to be explicitly unregistered, as all grammars of a session are automatically unregistered when the session terminates.

Note: Registering the same grammar (URI and parameters) multiple times without an explicit ID (i.e. the subsystem synthesizes an ID), will cause a new reference and ID to be generated for each tool invocation.

Inputs

Interaction

Identifier of the interaction.

Grammar URI

URI of the grammar to activate. The supported schemas and grammar formats are engine dependent. The URI may include a fragment identifier identifying a certain rule in the grammar. Support for fragments depends on the grammar type and (ASR) engine.

Grammar Type

Use this optional parameter to define the media type (MIME type) of the grammar referenced by the URL. If not specified, the recognition subsystem deduces the type from the file extension or the HTTP header. However, in general it's a good idea to specify the MIME type and not rely on the automatic deduction.

This parameter must be empty for `built-in` grammars.

If a type is specified that is different from the actual data, the tool will take the Invalid Type exit path. The following are the media types of the engine-agnostic grammar formats:

application/srgs	SRGS ABNF
application/srgs+xml	SRGS GrXML
application/x-jsgf	JSpeech

Grammar Mode

Mode of the referenced grammar. Thus, a grammar with mode "dtmf" is used for DTMF input. The tool will determine the grammar mode from data by default. An error will occur if it is not an engine agnostic grammar.

Note: A grammar cannot apply to more than one mode (thus, "voice dtmf" is not a valid mode).

Grammar Weight

The bias of this grammar in relation to other grammars, specified as a positive floating-point number. A value greater than 1.0 positively biases the grammar, and a value less than 1.0 negatively biases the grammar. The default value is 1.0. Not all engines support this and it is ignored if not supported.

Registration Mode

Controls when the grammar is sent to the ASR server:

Sync: If the grammar is not already on the ASR server, the grammar is sent to the server and compiled. The tool blocks until the registration succeeded. This most is best for testing as errors will be reported immediately.

Async: If the grammar is not already on the ASR server, the grammar is sent asynchronously and compiled in the background. The tool returns immediately. This most is best to register large grammars whose compilation may take a while and which aren't immediately used.

Lazy: The grammas is not sent to the ASR server (and no ASR port is allocated if the interaction doesn't yet have one). The grammar is sent to the ASR server the first time it is used for input. This is the most efficient mode, but grammar compilation errors may not be noticed until the first input is attempted with it.

Grammar ID

ID of the grammar. If not specified, the tool will synthesize one. The grammar IDs must only consist of alphanumeric characters as well as '\$' and '_'. IDs starting with '\$' are reserved for synthesized and other special IDs.

Override Duplicate ID

Specifies behavior if the session already has a grammar with this the specified GrammarID: Leave this box checked if you want pre-existing grammars to be overwritten by new grammars using the same ID. I.e., if a grammar with the specified ID has already been registered for this session, the old grammar will be un-registered and replaced with the new grammar.

Clear this box if you do not want older grammars to be overwritten. If this box is unchecked, then if a grammar with the specified ID has already been registered for this session, the tool will exit via the Failure exit path with error code "error.com.inin.grammar.id.duplicate".

Outputs

New Grammar ID

Grammar ID of the grammar. If a Grammar ID is specified as an Input, the same value is returned. If none is specified, this parameter returns a synthesized ID.

Error Code

If the registration fails, this output parameter contains an error code in the form of a VoiceXML event. This will be an empty string if no error occurred.

Error Text

If the registration fails, this output parameter will contain a simple textual description of what went wrong (e.g. reason for compilation error).

Exit Paths

Success

This path is taken if the grammar specified by the URL was successfully registered. This path may be taken even if the grammar is not actually valid as the engine may only fetch the grammar when it's actually used.

Failure

This path is taken if some other error occurred.

Reco Register Grammar String

This Reco tool compiles and registers a grammar specified as source text. The recognition subsystem will cache the compiled grammar or re-use an already compiled grammar transparently. The grammar will be active for all subsequent calls to [Reco Input](#) until the grammar is deactivated with [Reco Unregister Grammar](#), [Reco Unregister All Grammars](#), or [Reco Initialize](#).

The same grammar source may be registered with multiple IDs, as the IDs simply represent a "moniker" for the grammar. The behavior of registering a different grammar with an ID that already exists depends on the Override Duplicate ID parameter.

When the grammar is registered with Override Duplicate IDs and a grammar with the specified ID already exists, the existing grammar is replaced with the new grammar. Thus, the GrammarID now refers to the new grammar.

Grammars do not have to be explicitly unregistered, as all grammars of a session are automatically unregistered when the session terminates.

Note: Registering the same grammar (source and parameters) multiple times without an explicit ID (i.e. the subsystem synthesizes an ID), will cause a new reference and ID to be generated for each tool invocation.

Inputs

Interaction

Identifier of the interaction.

Grammar Source

Source code text of the grammar given as string.

Grammar Type

Use this optional parameter to define the media type (MIME type) of the grammar referenced by the URL. If not specified, the recognition subsystem deduces the type from the file extension or the HTTP header. However, in general it's a good idea to specify the MIME type and not rely on the automatic deduction.

This parameter must be empty for `built-in` grammars.

If a type is specified that is different from the actual data, the tool will take the Invalid Type exit path. The following are the media types of the engine-agnostic grammar formats:

application/srgs	SRGS ABNF
application/srgs+xml	SRGS GrXML
application/x-jsgf	JSpeech

Grammar Mode

Mode of the of referenced grammar. Thus, a grammar with mode "dtmf" is used for DTMF input. The tool will determine the grammar mode from data by default. An error will occur if it is not an engine agnostic grammar.

Note: A grammar cannot apply to more than one mode (Thus, "voice dtmf" is not a valid mode).

Grammar Weight

The bias of this grammar in relation to other grammars, specified as a positive floating-point number. A value greater than 1.0 positively biases the grammar, and a value less than 1.0 negatively biases the grammar. The default value is 1.0. Not all engines support this and it is ignored if not supported.

Registration Mode

Controls when the grammar is sent to the ASR server:

Sync: If the grammar is not already on the ASR server, the grammar is sent to the server and compiled. The tool blocks until the registration succeeded. This most is best for testing as errors will be reported immediately.

Async: If the grammar is not already on the ASR server, the grammar is sent asynchronously and compiled in the background. The tool returns immediately. This most is best to register large grammars whose compilation may take a while and which aren't immediately used.

Lazy: The grammas is not sent to the ASR server (and no ASR port is allocated if the interaction doesn't yet have one). The grammar is sent to the ASR server the first time it is used for input. This is the most efficient mode, but grammar compilation errors may not be noticed until the first input is attempted with it.

Grammar ID

ID of the grammar. If not specified, the tool will synthesize one. The grammar IDs must only consist of alphanumeric characters as well as '\$' and '_'. IDs starting with '\$' are reserved for synthesized and other special IDs.

Override Duplicate ID

Specifies behavior if the session already has a grammar with this the specified GrammarID: Leave this box checked if you want pre-

existing grammars to be overwritten by new grammars using the same ID. I.e., if a grammar with the specified ID has already been registered for this session, the old grammar will be un-registered and replaced with the new grammar.

Clear this box if you do not want older grammars to be overwritten. If this box is unchecked, then if a grammar with the specified ID has already been registered for this session, the tool will exit via the Failure exit path with error code "error.com.inin.grammar.id.duplicate".

Outputs

New Grammar ID

Grammar ID of the grammar. If a Grammar ID is specified as an Input, the same value is returned. If none is specified, this parameter returns a synthesized ID.

Error Code

If the registration fails, this output parameter contains an error code in the form of a VoiceXML event. This will be an empty string if no error occurred.

Error Text

If the registration fails, this output parameter will contain a simple textual description of what went wrong (e.g. reason for compilation error).

Exit Paths

Success

This path is taken if the grammar by the grammar text has been successfully compiled and activated.

Failure

This path is taken if some other error occurred.

Reco Register Inline Grammar

This Reco tool associates an inline grammar source with a language identifier. During execution, the tool will pick the grammar source based on the language of the call. This tool is therefore particularly useful to provide small grammars inline in the handler. Otherwise, this tool behaves like [Reco Register Grammar String](#).

One grammar may have a wildcard language identifier ("*"), which causes this grammar to be picked regardless of the interaction language. This is most useful for DTMF grammars, which are usually language independent. Matching the language IDs is done as follows: The grammars are ordered such that more specific language IDs are checked before the generic ones. Thus, "en-US" and "en-GB" are checked before "en." The language of the interaction matches the less specific IDs. Thus, an interaction with language "en-CA" will match a grammar whose language ID is "en." An interaction with the language "en" will match any language ID "en" or "en-X."

Inputs

Interaction

Identifier of the interaction.

Grammar Mode

Mode of the of referenced grammar. Thus, a grammar with mode "dtmf" is used for DTMF input. The tool will determine the grammar mode from data by default. An error will occur if it is not an engine agnostic grammar.

Note: A grammar cannot apply to more than one mode (Thus, "voice dtmf" is not a valid mode).

Grammar Type

Use this optional parameter to define the media type (MIME type) of the grammar referenced by the URL. If not specified, the recognition subsystem deduces the type from the file extension or the HTTP header. However, in general it's a good idea to specify the MIME type and not rely on the automatic deduction.

This parameter must be empty for `built-in` grammars.

If a type is specified that is different from the actual data, the tool will take the Invalid Type exit path. The following are the media types of the engine-agnostic grammar formats:

<code>application/srgs</code>	SRGS ABNF
<code>application/srgs+xml</code>	SRGS GrXML
<code>application/x-jsgf</code>	JSpeech

Grammar Weight

The bias of this grammar in relation to other grammars, specified as a positive floating-point number. A value greater than 1.0 positively biases the grammar, and a value less than 1.0 negatively biases the grammar. The default value is 1.0. Not all engines support this and it is ignored if not supported. Outputs

Registration Mode

Controls when the grammar is sent to the ASR server:

Sync: If the grammar is not already on the ASR server, the grammar is sent to the server and compiled. The tool blocks until the registration succeeded. This most is best for testing as errors will be reported immediately.

Async: If the grammar is not already on the ASR server, the grammar is sent asynchronously and compiled in the background. The tool returns immediately. This most is best to register large grammars whose compilation may take a while and which aren't immediately used.

Lazy: The grammas is not sent to the ASR server (and no ASR port is allocated if the interaction doesn't yet have one). The grammar is sent to the ASR server the first time it is used for input. This is the most efficient mode, but grammar compilation errors may not be noticed until the first input is attempted with it.

Grammar ID

Grammar ID of the grammar. If a Grammar ID is specified as an Input, the same value is returned. If none is specified, this parameter returns a synthesized ID.

Override Duplicate ID

Specifies behavior if the session already has a grammar with this the specified GrammarID: Leave this box checked if you want pre-existing grammars to be overwritten by new grammars using the same ID. I.e., if a grammar with the specified ID has already been registered for this session, the old grammar will be un-registered and replaced with the new grammar.

Clear this box if you do not want older grammars to be overwritten. If this box is unchecked, then if a grammar with the specified ID has already been registered for this session, the tool will exit via the Failure exit path with error code "error.com.inin.grammar.id.duplicate".

Outputs

Value

New Grammar ID value.

Code (event)

If the registration fails, this output parameter contains an error code in the form of a VoiceXML event. This will be an empty string if no error occurred.

Text (message)

If the registration fails, this output parameter will contain a simple textual description of what went wrong (e.g. reason for compilation error).

Exit Paths

Success

This path is taken if the grammar by the grammar text has been successfully compiled and activated.

Failure

This path is taken if some other error occurred.

Reco Register Preloaded Grammar

This Reco tool registers a preloaded grammar for use by this session. It corresponds to invoking the [Reco Register Grammar](#) tool with a URI of `x-inin-reco:preloaded/Name`.

Inputs

Interaction

Identifier of the interaction.

Name

Name of the preloaded grammar. The preloaded grammar must have been added previously in Interaction Administrator or the [Reco Add Preloaded Grammar](#) tool.

Weight

The bias of this grammar in relation to other grammars, specified as a positive floating-point number. A value greater than 1.0 positively biases the grammar, and a value less than 1.0 negatively biases the grammar. The default value is 1.0. Not all engines support this and it is ignored if not supported.

Grammar ID

ID of the grammar. If not specified, the tool will synthesize one. The grammar IDs must only consist of alphanumeric characters as well as '\$' and '_'. IDs starting with '\$' are reserved for synthesized and other special IDs.

Override Duplicate ID

Specifies behavior if the session already has a grammar with this the specified GrammarID. Leave this box checked if you want pre-existing grammars to be overwritten by new grammars using the same ID. I.e., if a grammar with the specified ID has already been registered for this session, the old grammar will be un-registered and replaced with the new grammar.

Clear this box if you do not want older grammars to be overwritten. If this box is unchecked, then if a grammar with the specified ID has already been registered for this session, the tool will exit via the Failure exit path with error code "error.com.inin.grammar.id.duplicate".

Outputs

New Grammar ID

Grammar ID of the grammar. If a Grammar ID is specified as an Input, the same value is returned. If none is specified, this parameter returns a synthesized ID.

Error Code

If the registration fails, this output parameter contains an error code in the form of a VoiceXML event. This will be an empty string if no error occurred.

Error Text

If the registration fails, this output parameter will contain a simple textual description of what went wrong (e.g. reason for compilation error).

Exit Paths

Success

The grammar by the grammar text has been successfully compiled and activated.

Failure

Some other error occurred.

Reco Release

This Reco tool requests a release of the recognition resources (ASR engine) and session. A handler can use this tool to explicitly release the resources when it no longer needs the ASR engine. This is merely considered as a (strong) hint to the recognition subsystem, but it may not honor it for some reason.

Irrespective of whether the ASR session is actually released or not, invoking this tool will un-register all grammars of the session.

Note: This tool does nothing if the interaction does not have an active recognition session. However in this case, it will still take the Success exit path.

Inputs

Interaction

Identifier of the interaction.

Exit Paths

Success

This path is taken if the request to release session is successfully processed.

Not Owner

This path is taken if the invoking handler is not the owner of the interaction.

Failure

This path is taken if an error has occurred.

Reco Session Active

This Reco tool checks whether the specified interaction has an active recognition session. Note that this does not necessarily mean that an ASR port is currently in use if the 'Lazy Port Allocation' checkbox is checked in the [Reco Initialize](#) tool. This tool just checks whether Reco Initialize has been called. Even if Reco Initialize has been called with 'Input Modes' 1 (DTMF), and thus no ASR session is active, this tool will return YES.

Inputs

Interaction

Identifier of the interaction.

Exit Paths

YES

There is currently a recognition session active for the specified interaction.

NO

The specified interaction does not have a recognition session.

Reco Set Inline Properties

This Reco tool sets multiple properties at once. The property names and their values are specified in the tool dialog.

Inputs

Interaction

Identifier of the interaction.

Outputs

Code (event)

This output parameter contains the error code if the tool failed.

Text (message)

Text accompanying the error code.

Exit Paths

Success

Value of the property was successfully retrieved.

Failure

Some other error occurred.

Reco Set Input Modes

This Reco tool sets the `Eic_RecoInputModes` interaction attribute to the specified value. If the 'Only supported modes' checkbox is checked, only those bits in the interaction attribute will be set whose associated input modes are currently enabled by the session. For example, if the 'Input Mode' argument is "dtmf voice," but the server doesn't support ASR or it was not enabled for the session of this interaction ('Reco Initialize'), the interaction attribute will be set to "dtmf."

If the interaction does not yet have a session ([Reco Initialize](#) has not yet been called), the `Eic_RecoInputModes` attribute will only be constrained by whether ASR is supported or not. Thus, calling this tool before calling `Reco Initialize` and subsequently calling `Reco Initialize` with undefined Input Modes has the same effect as calling `Reco Initialize` with the Input Modes parameter defined.

After `Reco Initialize` has been called, this tool is useful to temporarily disable the ASR (or DTMF) input as needed for all subsequent inputs (unless again overwritten by the [Reco Input](#) or [Reco Basic Input](#) tools).

Note: If the session was initialized without ASR (DTMF only), forcing an input mode that enables voice input will not cause an ASR session to be created. The interaction attribute will be changed and thus the prompts may change, but the input will still be limited DTMF. The `Reco Initialize` tool must be used to activate ASR for a session.

Inputs

Interaction

Identifier of the interaction.

Input Modes

Input modes to enable. This parameter is optional, and by default is set to modes enabled by [Reco Initialize](#) or the default of the recognition subsystem (if no session).

Only set modes supported for the interaction

This checkbox specifies how the input modes are set. It is checked by default and will thus only set supported modes. For example, if the argument is "dtmf voice" and ASR is not installed or has not been initialized for this session, the Eic_RecoInputModes attribute is set to "1."

Uncheck this box if you want the Eic_RecoInputModes attribute to be set to the value of the Input Modes argument, regardless of whether the specified input modes are supported.

Outputs

Old Input Modes

A space delimited list of names of the previous value(s) of the Eic_RecoInputModes attribute.

Error Code

This output parameter contains the error code if the tool failed.

Error Text

Text accompanying the error code.

Exit Paths

Success

This path is taken if the value has been set successfully.

None Set

This path is taken if the Eic_RecoInputModes attribute has been set to 0 because either the Input Modes argument is 0 or all modes are not supported (and the 'Only supported modes' checkbox is checked).

Failure

This path is taken if an error occurred.

Reco Set Property

This Reco tool sets the value of a recognition property. These properties are managed by the recognition subsystem and used to control ASR engine specific or extended functions.

Inputs

Interaction

Identifier of the interaction.

Property Name

Name of the property to modify. This string is case-sensitive.

Property Value

Value to set for the property.

Outputs

Error Code

This output parameter contains the error code if the tool failed.

Error Text

Text accompanying the error code.

Exit Paths

Success

This path is taken if the value of the property was successfully retrieved.

Failure

This path is taken if some other error occurred.

Reco Set Simulator Script State

This Reco tool changes the active state in the simulation script for an interaction. It also returns the <CustomData> child element if the <State> element has one.

Note: The state can be changed at any time, but changing the state while an input operation is active could lead to unexpected behavior.

Inputs

Interaction

Identifier of the simulated interaction.

New State

Identifier of the state in the simulation script ('id' attribute of the <State> element) to be activated. If this is an empty string, the first state will be activated.

Outputs

Custom Data

<CustomData> child element of the <State> element in the simulation script. NULL node if the <State> element does not have a <CustomData> child element.

Exit Paths

Success

This path is taken if the simulated interaction successfully queried.

Failure

This path is taken if there is an error changing the state. Common causes for this are if the state is invalid, the interaction has already been disconnected, or it's not a simulated interaction.

Reco Unregister All Grammars

This Reco tool unregisters all grammars registered on the session of the specified interaction.

Inputs

Interaction

Identifier of the interaction.

Outputs

Error Code

If the registration fails, this output parameter contains an error code in the form of a VoiceXML event. This will be an empty string if no error occurred.

Error Text

If the tool fails, this output parameter will contain a simple textual description of what went wrong (e.g. reason for compilation error).

Exit Paths

Success

This path is taken if all grammars were successfully unregistered.

Failure

This path is taken if an error occurred.

Reco Unregister Grammar

This Reco tool unregisters a grammar that was previously registered through the [Reco Register Grammar](#), [Reco Register Grammar String](#), or [Reco Register Inline Grammar](#) tool. The grammar is identified by its grammar ID. This tool simply tells the recognition subsystem that the grammar will not be used anymore and the associated resources can be released. You don't necessarily have to invoke this tool, as the recognition subsystem will collect the grammars when the session completes.

In general, it is not necessarily a good idea to explicitly unregister grammars that may be re-registered shortly again (e.g. in a subroutine handler that is invoked multiple times) and if they have an explicitly specified GrammarID as this defeats grammar registration optimizations done by the Reco Subsystem.

Inputs

Interaction

Identifier of the interaction.

Grammar ID

Space delimited list of identifiers of the grammar(s) to unregister.

Outputs

Error Code

If the tool fails, this output parameter contains an error code in the form of a VoiceXML event. This will be an empty string if no error occurred.

Error Text

If the registration fails, this output parameter will contain a simple textual description of what went wrong (e.g. reason for compilation error).

Exit Paths

Success

This path is taken if the grammar was successfully unregistered.

Failure

This path is taken if an error occurred.

Reco Verifier Abort Training

This Reco tool is used to discard the cumulative voiceprint data that has been collected during the current training session. The verifier database will not be affected.

Inputs

Interaction

Identifier of the interaction.

Outputs

Error Code

If the tool fails, this output parameter contains an error code in the form of a VoiceXML event. This will be an empty string if no error occurred.

Error Text

If the tool fails, this output parameter will contain a simple textual description of what went wrong.

Exit Paths

Success

The voiceprint data of the current training session has been discarded.

Failure

Some other error occurred. Consult the Error Code and Error Text outputs for details.

Reco Verifier Commit Training

This Reco tool is used to commit a verifier training session to the verifier database associated with the session. It will only succeed when a training session is currently active.

Inputs

Interaction

Identifier of the interaction.

Outputs

Error Code

If the tool fails, this output parameter contains an error code in the form of a VoiceXML event. This will be an empty string if no error occurred.

Error Text

If the tool fails, this output parameter will contain a simple textual description of what went wrong.

Exit Paths

Success

The current training session voiceprint data was committed to the verifier database.

Failure

Some other error occurred. Consult the Error Code and Error Text outputs for details.

Reco Verifier Ignore Last Utterance

This Reco tool is used to discard the last utterance received during a training or verification session. This is typically used when it is discovered that the last detected utterance doesn't match the expected utterance result. Another example would be if a 'help' grammar was specified so the user could receive additional help while in a training/verification session. If the Ignore Last Utterance tool wasn't used in this case, then the utterance for 'help' would count towards the voiceprint of the training/verification session and lead to an invalid voiceprint entry.

Inputs

Interaction

Identifier of the interaction.

Outputs

Error Code

If the tool fails, this output parameter contains an error code in the form of a VoiceXML event. This will be an empty string if no error occurred.

Error Text

If the tool fails, this output parameter will contain a simple textual description of what went wrong.

Exit Paths

Success

The last utterance spoken during a training or verification session is discarded and removed from the cumulative set of voiceprint analysis data.

Failure

Some other error occurred. Consult the Error Code and Error Text outputs for details.

Reco Wait For Grammars

This Reco tool can be used in conjunction with the asynchronous grammar registration mode to ensure that all grammars are ready before an actual recognition takes place and alternatively fall back to DTMF input.

Very large voice grammars may take quite a while to compile. The first call that registers that grammar would block in the [Reco Input](#) tool until the grammars are ready on the ASR server. Using this tool allows the handler to register several grammars asynchronously, go on to do other processing, return later to determine if the grammars are ready, and then decide whether to use DTMF or voice input.

If any of the specified grammars has been registered with the Lazy registration mode, they will be sent to the ASR server and the tool will wait for their registration.

DTMF grammar can be specified as arguments also, but DTMF grammars are always ready immediately after registration, regardless of registration mode.

Note: This tool is intended for advanced applications. As it adds some processing overhead, it should only be used if the application requires this kind of dialog processing. If at all possible, large grammars should be precompiled or preloaded.

Inputs

Interaction

Identifier of the interaction.

Grammars

Space delimited list of grammar IDs to check.

Max. Wait Time

Maximum amount of time to wait for the grammars to become available. The default wait time is five seconds. Enter a negative value if you want the tool to wait indefinitely.

Outputs

Pending Grammars

Space delimited list of grammars that are not yet ready.

Error Code

If the registration fails, this output parameter contains an error code in the form of a VoiceXML event. This will be an empty string if no error occurred.

Error Text

If the tool fails, this output parameter will contain a simple textual description of what went wrong (e.g. reason for compilation error).

Exit Paths

Ready

This path is taken if all specified grammars are ready for recognition.

Not Ready

This path is taken if one or more of the grammars is not yet ready.

Failure

This path is taken if an error occurred.

Reports

Report Tools

Every thirty minutes (or at some other interval you specify in Interaction Administrator), Queue Manger generates statistical information about the activity of each queue. This information includes the number of items that were in the queue and other statistical information about those items. For reporting purposes, the statistical information can be generated in several ways:

First, Queue Manager always generates statistical information for specified queues for every item.

Second, you can designate certain items as belonging to Stats Groups. Stats groups can contain items that can be dispersed among different queues. At the designated interval when other statistics are generated, Queue Manger will generate statistics for Stats Groups, even if those items reside on different queues. Stats Groups are a way for you to monitor a Group of calls or other items despite their location within the system.

Third, you can designate certain items as belonging to a Report Group. Report Groups are useful for reporting on a subset of the calls within the queue or Stats Group. When Queue Manger generates statistics for a queue, it additionally generates statistics for all the items belonging to a Report Group. An item can belong to one only Report Group.

Click on one of the tools below for more information on that tool:

[Advance Counter](#)

[Assign Report Group](#)

[Assign Stats Group](#)

[Get Nth Period Statistics Report Data](#)

[IVR Event Notify](#)

[IVR_EnterLevel](#)

[IVR_ExitIVR](#)

[Log Tools](#)

[Query Counter](#)

[Query Report Group](#)

[Query Stats Group](#)

[Remove Stats Group](#)

[Report Email](#)

[Report Export File](#)

[Report HTML Export](#)

[Report Print](#)

Log Tools

Each log tool writes one or more pieces of information to a report log with a corresponding name. Each input parameter in a log tool corresponds to a column defined in the report log in Interaction Administrator.

When you install CIC Server, the following report logs are configured automatically:

- **Account Code Configuration Log**
This log contains a mirror image of the account codes as configured in CIC.
- **Agent Activity Log**
This log contains details of individual status changes. It is updated each time an agent manually changes phone status (e.g. "at lunch", "available", etc.).
- **Agent Queue Statistics Interval Log**
The Agent Queue Statistics Interval log was deprecated in CIC 4.0.
This log contains details inserted from the Queue Period Statistics monitor handlers.
- **Fax Envelope History Log**
This log contains statistics on faxes sent through CIC.
- **IC Change Notification Log**
This log contains details of the change notifications transmitted by the Admin Server.
- **Interaction Administrator Change Notification Log**
This log contains details about all changes made using the Interaction Administrator program.
- **Interaction Custom Attributes Record Log**
- **Interval Line Group Statistics Log**
This log contains interval statistics for line groups configured for reporting in CIC.
- **Interval Line Statistics Log**
This log contains interval statistics for lines configured for reporting in CIC.
- **IVR History Log**
This log contains the history of each interaction passed through IVR nodes configured in CIC's Interaction Attendant.
- **IVR Interval Log**
This log contains the statistics of each IVR node configured in CIC's Interaction Attendant. This data is collected periodically based on server parameters.
- **Line Configuration Mirror Log**
This log contains a mirror image of the lines as configured in CIC.
- **Line Group Configuration Log**
This log contains a mirror image of the line groups as configured in CIC.
- **Line Group to Lines Relationship Mirror Log**
This log contains a mirror image of the line group to lines relationship as configured in CIC.
- **Statistics Group Interval Log**
The Statistics Group Interval Log was deprecated in CIC 4.0.
This log contains details inserted by the Queue Period Statistics monitor handlers. It is similar to Agent Queue Statistics Interval Log except that the agent values are not present because statistics groups are not associated with agents.
- **User to Workgroup Relationship Log**
This log contains a mirror image of the users and their workgroups as configured in Interaction Administrator.
- **Workgroup Queue Statistics Interval Log**
The Workgroup Queue Statistics Interval log was deprecated in CIC 4.0.
This log contains details inserted from the Queue Period Statistics monitor handlers. It is similar to Agent Queue Statistics Interval Log, but contains information on workgroup/queue for which there was activity, or agents logged in the system.
- **WrapUp Statistics Log**
This log contains statistics for each wrap-up code as configured in CIC.

Note: Because log tools send data to the log server asynchronously, error messages resulting from tool failure are written to the log file.

Report log tools fail if the tool is out of sync with the actual report log defined in Interaction Administrator. If you use a custom column in a report log, you must update any handlers that currently use the corresponding log tool to write information to that log.

See the PureConnect Reporting Technical Reference, the Advanced Reporting Technical Reference, and the PureConnect Data Dictionary for more information on logs, log tools, logging handlers, custom columns, and reporting.

Advance Counter

Statistics counters gather information about a particular call (telephone or chat) as it moves across queues. Two of these statistics counters are always in effect for every call: ConferenceCount and TransferCount. **ConferenceCount** is incremented automatically for a call when that call is added to a conference. **TransferCount** is incremented automatically for a call when the call is transferred by either a successful blind transfer or a successful consult transfer.

Use this Reports tool to create or increment your own custom statistics counters for ACD-processed interactions. For example, if you want to know how many times a call has entered a certain workgroup, create a statistics counter for that call using this tool. Any time a call enters that workgroup, use this counter to increment that statistics counter. When Queue Manager generates statistical information for that queue, it will also generate the information for any custom statistics.

Note: This tool only works for interactions in ACD processing.

Inputs

Call Id

ID of the call to be assigned a counter or to have a counter advanced.

Statistics Counter Identifier

The name of the statistics counter.

Exit Paths

Next

This step always takes the Next exit path.

Assign Report Group

This Reports tool assigns a call to a report group. A call can be a member of only one report group at a time. Report groups cannot be assigned once the call has been moved to a queue. See the [Report Tools](#) overview for more information on Report tools.

Inputs

Call Id

ID of the call to be assigned to a report group.

Report Group Name

The name of the report group to which the call will be assigned.

Exit Paths

Next

This step always takes the Next exit path.

Assign Stats Group

This Reports tool assigns a telephone call or chat session to a Stats group.

At regular intervals (every 30 minutes, or some other interval you configure in the QueuePeriodStatisticsInterval server parameter in Interaction Administrator), Queue Manager generates statistical information about calls that have changed states in designated queues. This statistical information eventually ends up in a database and is used for reporting purposes. In addition to the statistics generated for the queues, Queue Manager also generates statistics for all the calls that belong to a statistics group.

Statistics groups are a way of categorizing the calls for which statistical information is being generated. Statistics generated for calls in statistical groups is limited to information about the calls, and does not include the additional agent information generated for queues. Statistics information for statistics groups is grouped together, even if these calls reside on different queues. A call can belong to more than one statistics group.

As an example of how statistics groups work, imagine that your call center receives calls on three separate toll free numbers. You want to determine how well your agents are processing these calls, so you need a way of organizing the statistics information generated by Queue Manager. Each time a call enters SystemIVRWorkgroup, you assign that call to one of three statistics groups depending upon which toll free number the caller dialed. When Queue Manager generates statistics information about all of the calls in CIC, it also lists statistics for the groups of calls belonging to each statistics group. In other words, Queue Manager generates statistics info for all of the calls, then for the three separate statistics groups.

Inputs

Call Id

ID of the call to be assigned to a stats group.

Statistics Group Name

The name of the statistics group.

Exit Paths

Next

This step always takes the Next exit path.

Get Nth Period Statistics Report Data

This Reports tool retrieves a list of information at a position you specify from an array of statistical information passed in by the [Queue Period Report Statistics Initiator](#).

Note: This tool is deprecated and may be removed in a future release of CIC.

IVR Event Notify

This tool generates IVR events for each incoming call interaction. CX Insights uses the IVR event data as part of analytics. Place this tool in a handler such that it is called every time an interaction moves to a new node in an IVR. When called, this tool sets various interaction attributes and IVR level values.

Note: "Level" in IVR refers to the interaction's use in Interaction Attendant's node tree.

Inputs

Interaction Id

Integer representing the unique identifier for the interaction.

Level Number

Integer indicating level of the menu being entered. These levels have a value of 0 through 6.

Level in this sense does not refer to the actual location or tier in the menu structure. Level refers to the function performed at that location. This tool uses the following level settings:

0	Server
1	Profile
2	Schedule
3	Application/Complex Operation
4	Menu
5	Task
6	Exit Path

See the online help for Interaction Attendant for more information on IVR nodes.

Level Name

Name of the level being entered.

Level Hierarchy

Cumulative level information from initial level to the most recent entered level.

Level Id Hierarchy

Cumulative level Id (unique identifier for IVR node in Interaction Attendant) parallel to level hierarchy.

Completion Type

The final interaction resolution, either Completed or Transferred.

Queue

The queue is the destination for the call.

Action

Indicates NoInput or NoMatch action at the entered level.

Outputs

Retval

Integer indicates if IVR event is successfully generated.

Exit Paths

Success

The success exit path is taken if IVR event is successfully generated.

Failure

The failure exit path is taken if the operation fails.

IVR_EnterLevel

This tool is used in conjunction with [IVR_ExitIVR](#) to track an interaction's movements through an IVR. For accurate reporting, this tool should be placed in a handler such that it is called every time an interaction moves to a new node in an IVR. When called, this tool sets the IVR level's value for the interaction. "Level" in IVR refers to the interaction's use in Interaction Attendant's node tree. When this tool is called, it sets the level value for the interaction on the IC_ATTR_IVR_CURRENT_LEVELS interaction attribute. If IC_ATTR_IVR_CURRENT_LEVELS was already set, this tool will also set the IC_ATTR_IVR_PREVIOUS_LEVELS attribute appropriately.

Notes: What is referred to as "level" by these tools is also referred to as "Nodes" in the online help for Interaction Attendant. "Level" in this sense refers only to the function performed at that tier of the IVR, and does not necessarily have any relation to the linear progression of tiers in the IVR structure. See the help documentation for that application for more information.

IVR is considered to have already started when a level is set. If a level name is set, and there is no value for the levels above, they will be assumed to be "-" for unspecified for the purposes of further IVR processing.

Inputs

CallID

Integer representing the unique identifier for the interaction. Use the [Call Id To Integer](#) tool to convert the Call Id to an Integer to pass into this tool.

IVR Level Name

Name of the level being entered. This string is limited to 50 characters. If a name is greater than 50 characters in length, it will be truncated at 50.

IVR Level

Integer indicating Level of the menu being entered. These levels have a value of 1 through 5.

As stated above, level in this sense does not refer to the actual location or tier in the menu structure. Level refers to the function performed at that location. The Interaction Attendant handlers make the following level settings by default:

1	Profile
2	Schedule
3	Application/Complex Operation
4	Menu
5	Task

See the online help for Interaction Attendant for more information on IVR nodes.

Exit Paths

Success

This tool always takes the Success exit path.

IVR_ExitIVR

This tool is used in conjunction with [IVR_EnterLevel](#) to track an interaction's movements through an IVR. For accurate reporting, this tool should be called whenever an interaction leaves in IVR.

Note: this tool and IVR_EnterLevel report IVR activity, not interaction activity. If a call leaves IVR for any reason, then subsequently re-enters IVR, it will be reported as a separate instance.

Inputs

Call ID

Integer representing the unique identifier for the interaction. Use the [Call Id To Integer](#) tool to convert the Call Id to an Integer to pass into this tool.

IVR Exit Code

Positive integer indicating the reason that the interaction left IVR. By default, the number one (1) is used to denote an abandoned call. Other positive values may be assigned for custom IVR routing if desired.

IVR Exit Path

This string denotes the destination of the interaction when it exits IVR. In this string, the colon (:) is a reserved character used to truncate the exit path. I.e., anything after a colon will be dropped from the string. Thus, interactions with an exit path string of "Workgroup:Marketing" or "UserQueue:MarkM" will be read as "Workgroup" or "UserQueue" respectively.

Note: Because of the way this string is truncated, it is **not** possible to use a full directory path as the exit path. A string such as "c:\ic\admin" will be truncated to simply "c".

Exit Paths

Success

This tool always takes the Success exit path.

Logging Custom Passthrough

Executes a string containing an ODBC SQL statement. CIC Logging Server executes this string on the same database connection that is used for all of our log info. This tool allows you to send an Insert statement through PMQ queues to the CIC Logging and have the insert happen in the same server and database. This lets you reliably send your own data to custom logging tables with the same reliability as used for CIC data.

To use the Logging Custom Passthrough tool, set **9999 - Custom Passthrough** to **Yes** in the Report Logs subcontainer under System Configuration in Interaction Administrator. By default, **9999-Custom Passthrough** is set to **No**.

This string is not error checked.

Inputs

SQL Text to be executed on Logging Server

The string containing the SQL statement to execute.

Exit Paths

Success

This tool takes the Success exit path unless a failure condition occurred (see failure conditions below).

Failure

This tool takes the Failure exit path if it was unable to send the command due to PMQ problems. Since this tool is asynchronous, any database errors caused by an invalid statement or caused by some other type of error will occur on the CIC logging machine and be logged with CIC logging errors.

Purge Log Records

Purge Log Records deletes data in a log file. You specify the date, and this tool deletes all records created before that date. This tool allows you to schedule (using a handler started with the Timer initiator) the deletion of old information.

Note: This tool works only if your logs are stored in a Jet database. This tool does not work with CSV databases.

Inputs

Log ID

The Log ID from the Report Log container in Interaction Administrator. Although this parameter takes a string, enter the Log ID number.

Cut Off Time

This tool deletes all records created before the date you specify in this parameter.

Exit Paths

Success

This step takes the Success exit path if the records are deleted.

Failure

This step takes the Failure exit path if the Log ID is invalid.

Query Counter

This Reports tool returns the value of a statistics counter. See [Advance Counter](#) for more information on statistics counters.

Inputs

Call Id

ID of the telephone call or chat session that has the associated statistics counter.

Statistics Counter Identifier

The name of the statistics counter.

Outputs

Statistics Counter Value

The current value of the statistics counter.

Exit Paths

Next

This step always takes the Next exit path.

Query Report Group

This Reports tool returns the name of the report group to which a call belongs. A call can be a member of only one report group at a time. See the [Report Tools](#) overview for more information on Report tools.

Inputs

Call Id

ID of the call to be queried.

Outputs

Report Group Name

The name of the report group to which the call belongs.

Exit Paths

Next

This step always takes the Next exit path.

Query Stats Groups

This Reports tool returns a list of the statistics groups to which a call has been assigned. See the documentation for [Assign Stats Group](#) for more information on stats groups.

Inputs

Call Id

ID of the telephone call or chat session to be queried.

Outputs

List of Statistics Groups

A list of strings containing the names of any statistic groups to which the call belongs.

Exit Paths

Next

This step always takes the Next exit path.

Remove Stats Group

This Reports tool removes a telephone call or chat session from a stats group. See the documentation for [Assign Stats Group](#) for more information on stats groups.

Inputs

Call Id

ID of the telephone call or chat session to be removed from a stats group.

Statistics Group Name

The name of the statistics group.

Exit Paths

Next

This step always takes the Next exit path.

Report E-mail

Note:

In 2015 R2 and later, in order to schedule reports, users must have access to ICBM and have the Interaction Supervisor Plug-in: Historical Reporting license. This is a new requirement, as of 2015 R2. Additionally, the Report E-mail Tool can no longer be used for custom report scheduling operations. `ReportingExecutable.exe` along with the Shell Execute Command tool must be used. The Report E-mail Tool only works with Crystal Reports 9; it will not work with Crystal Reports 2013 (2015 R2, and later). For more information, see "Report e-mail tool" in the *Reporting Technical Reference*, and also see "Appendix C: Run an Interaction Reporter report or a Report Assistant report using command line parameters" in the *CIC Scheduled Reports Installation and Configuration Guide*.

This Reports tool exports the report in the desired format, and then sends the file to the Mail To, and Carbon Copy list. All temp files created in the process will be cleaned up automatically.

Mail addresses can be complicated. The actual emailing of the file is done using the Interaction Processor (IP) email tools. Thus this tool has all of the email restrictions and implementation details of those email tools. Mail To list must conform to addresses understood by the IP email tools. Currently the most confusing aspect of this is that MS Mail, and Outlook take user ID, or names and convert them internally to MS Mail addresses. This must be done explicitly on the mail list. For example GregC or Cunningham, Greg at Genesys is really `MS:/I3/I3HOME/GREGC` or `CCMAIL:Cunningham, Greg at I3-Home`, or `SMTP:GregC@genesys.com`, or `X400:c=US;a=;p=i3;o=I3-Home;s=GregC;`. These addresses can be separated with a semicolon to form a list of recipients.

Note:

`I3RunCrReport.exe` will auto-register as the processor of the RPT file extension. This will allow a RPT file to be opened and previewed simply by executing a shell open command on the file. A full copy of Crystal Reports is not required for a recipient to preview a RPT file. They need only install printing support, and execute `I3RunCrReport.exe` once to register the extension.

Inputs

Report Name

This is the report definition that will be used to processes the requested report. It must be an Identifier Name of a report configured in Interaction Administrator, and not the Display Name.

Export Type

The format of the resulting exported report. Some of the formats you can type here are:

- Microsoft Word for Windows
- Microsoft Excel 2.1
- Microsoft Excel 3.0
- Microsoft Excel 4.0
- Microsoft Excel 5.0
- Microsoft Excel 5.0 Extended
- Rich Text Format
- Crystal RPT
- Text
- Tabbed Text
- Paginated Text
- Comma Separated Text
- Tab Separated Text
- Character Separated Text
- Records
- DIF

Mail To

Specifies the recipient of the emailed report. You must specify the full internet address (`JohnD@YourOrg.com`), a Microsoft Exchange address (`EX:/o=I3/oe.....cn=JohnD`), or a IBM Notes/Domino address (`NOTES:CN=JohnDoe/OU=Indy/O=Genesys`). If you leave the To and CC parameters empty, the message is sent to the address specified in the Unaddressed Mail Recipient [server parameter](#).

Note:

In Directory Services, each user's Exchange address is stored in that user's User key in the emailAddress attribute.

Carbon Copy

Specifies who should receive a copy of this email. You must specify the full internet address (`JohnD@YourOrg.com`), a Microsoft Exchange address (`EX:/o=I3/oe.....cn=JohnD`), or a IBM Notes/Domino address (`NOTES:CN=JohnDoe/OU=Indy/O=Genesys`). Separate multiple email addresses with a semicolon.

Subject

Any text you want to appear in the email's subject line. (optional parameter)

Message

Any text you want to appear in the email's body. (optional parameter)

Boolean Parameter Names

This parameter takes a List of String value.

See [Specifying Report Parameters](#) for more information and examples.

Boolean Parameter Values

This parameter takes a List of Boolean value.

See [Specifying Report Parameters](#) for more information and examples.

String Parameter Names

This parameter takes a List of String value.

See [Specifying Report Parameters](#) for more information and examples.

String Parameter Values

This parameter takes a List of String value. The value you enter at a certain position must correspond with the Name value in the same position in the String Parameter Name parameter above.

See [Specifying Report Parameters](#) for more information and examples.

DateTime Parameter Names

This parameter takes a List of String value.

See [Specifying Report Parameters](#) for more information and examples.

DateTime Parameter Values

This parameter takes a List of DateTime value. The value you enter at a certain position must correspond with the DateTime Name value in the same position in the DateTime Parameter Name parameter above.

If only the date portion of the DateTime value is needed, or if only the time portion of the DateTime value is needed, this tool will only use that portion of the DateTime value.

See [Specifying Report Parameters](#) for more information and examples.

Number Parameter Names

This parameter takes a List of String value.

See [Specifying Report Parameters](#) for more information and examples.

Number Parameter Values

This parameter takes a List of Numeric value. The value you enter at a certain position must correspond with the Numeric Name value in the same position in the Numeric Parameter Names parameter above.

See [Specifying Report Parameters](#) for more information and examples.

Special Export Options

Depending on export type, any of the following values separated by a semicolon. [Click here for more information on Special Export Options](#):

- LinesPerPage=n
- UseReportNumberFormat=0 or 1
- UseReportDateFormat=0 to n, where n is the constant column width. Default is 9.
- StringDelimiter={character}
- FieldDelimiter={character}
- ExcelColumnHeadings=0 or 1
- ExcelConstColWidth=0 or 1
- ExcelTabularFormat=0 or 1
- ExcelBaseAreaType={PAGEHEADER, PAGEFOOTER, REPORTHEADER, REPORTFOOTER, GROUPHEADER, GROUPFOOTER}
- ExcelBaseArea=n where n is a group number and ExcelBaseAreaType=GROUPHEAER or GROUPFOOTER
- FirstPageNo=n where n is the first page number of the document. This option only applies when exporting .pdf or .rtf files.
- LastPageNo=n where n is the last page number of the document. This option only applies when exporting .pdf or .rtf files.

Exit Paths

Success

The Success exit path is taken if the report was generated successfully.

Failure

The Failure exit path is taken if the report failed. This can occur if one of the parameter names or values requested by the report was not supplied in your input values.

Report Export File

This Reports tool runs one of the Crystal reports defined in the Interaction Administrator Reports container. The output of this report is converted to a document type and saved in a folder you specify.

Inputs

Report Name

This is the report definition that will be used to process the requested report. It must be a Report Name (Identifier Name) of a report configured in Interaction Administrator, and not the Display Name of the report.

Export Type

Export format to be used for output of report. HTML Export types are excluded from this list. Valid types include:

- Acrobat Format (PDF)
- Character-separated values
- Comma-separated values (CSV)
- Crystal Reports (RPT)
- Crystal Reports 7.0 (RPT)
- Data Interchange Format (DIF)
- Excel 97-2000
- Excel 97-2000 (Data only)
- ODBC – dBASE Files
- ODBC – dBASE Files – Word
- ODBC – EIC IA Export
- ODBC – EIC_Tables
- ODBC – Excel Files
- ODBC – FoxPro Files – Word
- ODBC – LocalServer
- ODBC – MS Access Database
- ODBC – Visual FoxPro Database
- ODBC – Visual FoxPro Tables
- Paginated Text
- Record style (columns of values)
- Report Definition
- Rich Text Format
- Tab-separated text
- Tab-separated values
- Text
- Word for Windows document
- XML

Note: Excel versions 5.0, 7.0, and 8.0 are also supported, but are mapped automatically to Excel 97-2000. Extended versions are mapped to Excel 97-2000 (Data Only).

Export File Path and Name

The file name and location to which the report will be exported.

Note: Do not specify the file name extension.

Boolean Parameter Names

This parameter takes a List of String value.

See [Specifying Report Parameters](#) for more information and examples.

Boolean Parameter Values

This parameter takes a List of Boolean value.

See [Specifying Report Parameters](#) for more information and examples.

String Parameter Names

This parameter takes a List of String value.

See [Specifying Report Parameters](#) for more information and examples.

String Parameter Values

This parameter takes a List of String value. The value you enter at a certain position must correspond with the Name value in the same position in the String Parameter Name parameter above.

See [Specifying Report Parameters](#) for more information and examples.

DateTime Parameter Names

This parameter takes a List of String value.

See [Specifying Report Parameters](#) for more information and examples.

DateTime Parameter Values

This parameter takes a List of DateTime value. The value you enter at a certain position must correspond with the DateTime Name value in the same position in the DateTime Parameter Name parameter above.

If only the date portion of the DateTime value is needed, or if only the time portion of the DateTime value is needed, this tool will only use that portion of the DateTime value.

See [Specifying Report Parameters](#) for more information and examples.

Number Parameter Names

This parameter takes a List of String value.

See [Specifying Report Parameters](#) for more information and examples.

Number Parameter Values

This parameter takes a List of Numeric value. The value you enter at a certain position must correspond with the Numeric Name value in the same position in the Numeric Parameter Names parameter above.

See [Specifying Report Parameters](#) for more information and examples.

Special Export Options

Depending on export type, any of the following values separated by a semicolon. [Click here for more information on Special Export Options](#):

- LinesPerPage=n
- UseReportNumberFormat=0 or 1
- UseReportDateFormat=0 to n, where n is the constant column width. Default is 9.
- StringDelimiter={character}
- FieldDelimiter={character}
- DSN
- UID
- PWD
- TABLE
- ExcelColumnHeadings=0 or 1

- ExcelConstColWidth=0 or 1
- ExcelTabularFormat=0 or 1
- ExcelNoWorksheetFunc
- ExcelBaseAreaType={PAGEHEADER, PAGEFOOTER, REPORTHEADER, REPORTFOOTER, GROUPHEADER, or GROUPFOOTER}
- ExcelBaseAreaGroupNum=n where n is a group number and ExcelBaseAreaType=GROUPHEAER or GROUPFOOTER
- FirstPageNo=n where n is the first page number of the document. This option only applies when exporting .pdf or .rtf files.
- LastPageNo=n where n is the last page number of the document. This option only applies when exporting .pdf or .rtf files.

Exit Paths

Success

The Success exit path is taken if the report was generated successfully.

Failure

The Failure exit path is taken if the report failed. This can occur if one of the parameter names or values requested by the report was not supplied in your input values.

Report HTML Export

This Reports tool creates an HTML export file that contains a single running body of the report with a single footer, and a single header section. Obviously this type of export will only be applicable to certain applications.

The HTML Directory and File name are used to create a directory, with an HTML file contains the body of the report. Also placed into the directory are any of the related JPG files that must be generated to support graphics in the report. The directory used must be unique to the report, as this is the only way to guarantee that these supporting JPG files will not collide with JPG files from other exports.

Inputs

Report Name

This is the report definition that will be used to processes the requested report. It must be an Identifier Name of a report configured in Interaction Administrator.

HTML Output Type

The format of the HTML to be generated. Valid choice include:

- HTML 3.2
- HTML 4.0 (DHTML)

HTML Directory and File Name

File name with path. The exact intent is dependent on export type. HTML export types will use the file name as the base name for the HTML, and will put any JPG file in the associated directory with the file

If no path is supplied, the file will be placed in the CIC work directory if one exists, or the temp directory, or the root of the C: drive.

Boolean Parameter Names

This parameter takes a List of String value.

See [Specifying Report Parameters](#) for more information and examples.

Boolean Parameter Values

This parameter takes a List of Boolean value.

See [Specifying Report Parameters](#) for more information and examples.

String Parameter Names

This parameter takes a List of String value.

See [Specifying Report Parameters](#) for more information and examples.

String Parameter Values

This parameter takes a List of String value. The value you enter at a certain position must correspond with the Name value in the same position in the String Parameter Name parameter above.

See [Specifying Report Parameters](#) for more information and examples.

DateTime Parameter Names

This parameter takes a List of String value.

See [Specifying Report Parameters](#) for more information and examples.

DateTime Parameter Values

This parameter takes a List of DateTime value. The value you enter at a certain position must correspond with the DateTime Name value in the same position in the DateTime Parameter Name parameter above.

If only the date portion of the DateTime value is needed, or if only the time portion of the DateTime value is needed, this tool will only use that portion of the DateTime value.

See [Specifying Report Parameters](#) for more information and examples.

Number Parameter Names

This parameter takes a List of String value.

See [Specifying Report Parameters](#) for more information and examples.

Number Parameter Values

This parameter takes a List of Numeric value. The value you enter at a certain position must correspond with the Numeric Name value in the same position in the Numeric Parameter Names parameter above.

See [Specifying Report Parameters](#) for more information and examples.

Exit Paths

Success

The Success exit path is taken if the report was generated successfully.

Failure

The Failure exit path is taken if the report failed. This can occur if one of the parameter names or values requested by the report was not supplied in your input values.

Report Print

This Reports tool is used to output to a printer, or to file, via the printer drivers. A blank printer name will output to the default printer. The only input parameters that you need to provide for this tool are those that are User Defined in Interaction Administrator. All System Defined and fixed-value parameters will be passed in automatically.

Notes: If printing to file, no automated cleanup of the files will take place. Either automated or manual cleanup routines are required to remove any files after they have been used.

When CIC runs as a service, care must be taken to insure that the printers being used are accessible by the service. Services either must be given specific user privileges, or devices need to have printing rights granted to everyone to insure that printing is possible. Also the full path to the printer as seen on the Printer Properties/Security/Permissions will likely have to be used unless the printer is local.

Inputs

Report Name

This is the report definition that will be used to processes the requested report. It must be an Identifier Name of a report configured in Interaction Administrator, and not the Display Name.

Printer Name

Printer name used for output. A UNC path may be required to correct address the printer if it is a network printer. If no printer name is supplied, the default printer will be used.

Number of Copies

Tells print engine number of copies to print. Value defaults to one if nothing is supplied. The range is 0 to 65,000.

Starting Output Page

The starting page of the report. If nothing is specified, 1 is used.

Ending Output Page

The ending page of the report. If nothing is specified, the last page of the report is used.

Collate Flag

Set this value to "true" if you want to collate. Set this value to "false" if you do not want the report collated. If you leave this value empty, the default value is true.

Boolean Parameter Names

This parameter takes a List of String value.

See [Specifying Report Parameters](#) for more information and examples.

Boolean Parameter Values

This parameter takes a List of Boolean value.

See [Specifying Report Parameters](#) for more information and examples.

String Parameter Names

This parameter takes a List of String value.

See [Specifying Report Parameters](#) for more information and examples.

String Parameter Values

This parameter takes a List of String value. The value you enter at a certain position must correspond with the Name value in the same position in the String Parameter Name parameter above.

See [Specifying Report Parameters](#) for more information and examples.

DateTime Parameter Names

This parameter takes a List of String value.

See [Specifying Report Parameters](#) for more information and examples.

DateTime Parameter Values

This parameter takes a List of DateTime value. The value you enter at a certain position must correspond with the DateTime Name value in the same position in the DateTime Parameter Name parameter above.

If only the date portion of the DateTime value is needed, or if only the time portion of the DateTime value is needed, this tool will only use that portion of the DateTime value.

See [Specifying Report Parameters](#) for more information and examples.

Number Parameter Names

This parameter takes a List of String value.

See [Specifying Report Parameters](#) for more information and examples.

Number Parameter Values

This parameter takes a List of Numeric value. The value you enter at a certain position must correspond with the Numeric Name value in the same position in the Numeric Parameter Names parameter above.

See [Specifying Report Parameters](#) for more information and examples.

Print File

Forces output to be printed to this file. This can be used to create PostScript output or to print to file from the Fax Printer to create a file that can be processed for faxing.

If no path is supplied, the file will be placed in the CIC work directory if one exists, or the temp directory, or the root of the C: drive.

Exit Paths

Success

The report was generated successfully.

Failure

The report failed. This can occur if one of the parameter names or values requested by the report was not supplied in your input values.

REST

Array Builder

This REST tool accepts a list of values to build an array object of values. This tool builds the name/value pairs and inner JSON alphabetically by name. For more information, refer to the [Interaction Designer REST APIs Developer's Guide](#) in the PureConnect Documentation Library.

Inputs

Values List

The list of values.

Value Type List

The parallel list of data type for each value in the Values List parameter. The length of this list must match the length of the list in the Values List parameter.

Outputs

Array String

The variable to store the output.

Exit Paths

Success

This path is taken if the operation is successful.

Failure

This path is taken if the operation fails.

List Length Not Equal

This path is taken if the lengths of the input lists are not equal.

Value Type Mismatch

This path is taken if a value cannot be converted to the listed data type.

Invalid Value Type

This path is taken if a data type provided by the Value Type List parameter does not match an accepted data type (for example, string, bool, double, float, int, array, or object).

Array Parser

This REST tool outputs a list of values from an array. This tool parses the name/value pairs and inner JSON alphabetically by name. For more information, refer to the [Interaction Designer REST APIs Developer's Guide](#) in the PureConnect Documentation Library.

Inputs

Array String

A string representation of an array.

Outputs

Values List

The list of parsed values.

Value Type List

The parallel list of the data type for each value in the Values List parameter. Data types can include: string, bool, double, array, object (JSON), and null. These data types correspond to JSON data types: string, Boolean, number, array, object (JSON), and null. Integer and floating-point output as a double type (corresponding to JSON's number type).

Exit Paths

Success

This path is taken if the operation is successful.

Failure

This path is taken if the operation fails.

Array Parse Failure

This path is taken if the ION library cannot parse the string in the Array String parameter. This can result from a malformed array or malformed elements which may include another array or JSON object.

Bearer Token Request

This REST tool is used to obtain an OAuth 2.0 access bearer token for a client credentials grant or a password grant. This tool returns a parsed token from JSON that you can use in the Bearer Token parameter for the REST HTTP Request tool. The server response must be JSON. Use the client credentials grant for PureCloud APIs and the password grant for Salesforce.com APIs.

This tool:

- Expects the certificate file to exist in a specific directory: I3\IC\Certificates\REST. Create the I3\IC\Certificates\REST directory if it does not exist. The I3\IC\Certificates\REST directory is not created by default.

Use the CertTrustU.exe to install a client certificate. The certificate must be in PEM format. For more information, see [PureConnect Security Features Technical Reference](#) in the PureConnect Documentation Library.

- Supports many concurrent requests.
- Does not use Windows Certificate store to check for validity of the certificate and uses openssl to do the verification.

To obtain more information on any failure, check logs of Interaction Processor or the response body and headers for failure information.

For more information, refer to the [Interaction Designer REST APIs Developer's Guide](#) in the PureConnect Documentation Library.

Inputs

URL

The URL of the request.

Proxy Uri

(Optional) String URI of the HTTP proxy to use for REST calls. This parameter supports the use of a forward proxy to retrieve content from the origin (API) server. This parameter must include the protocol used. For example: `http://env4-revproxy1.ininlab` or `http://192.168.1.10`.

ID/Key

The ID or key required for client credentials grant and password grant. Do not encode.

Secret

Secret required for client credentials grant and password grant. Do not encode.

User Name

(Optional) Username required for password grant.

Password

(Optional) Password required for password grant.

Grant Type

(Optional) Specify whether to use a grant type of "client_credentials" or "password". If this parameter contains an empty string, the default "client_credentials" is used.

Credentials In POST Body ?

(Optional) Set to true to output the credentials in the body of the REST call. Set to false to encode the credentials in the header. You must set this parameter to true for Salesforce.com API compatibility.

Content Type

The content type to add to the header of the REST call. If this parameter contains an empty string, the default (`application/x-www-form-urlencoded`) is used.

Request Timeout [s]

Maximum time in seconds to wait for request to be sent and response received before terminating the REST call request. Default: 10 seconds.

Client SSL Certificate

(Optional) Specify whether the server certificate validates against a client certificate to establish the HTTPS connection (installed by using `CertTrustU.exe` in `I3\IC\Certificates\REST`). Create the `I3\IC\Certificates\REST` directory if it does not exist. The `I3\IC\Certificates\REST` directory is not created by default.

Ignore Unknown SSL Certificate Authority

This parameter applies only if the URL protocol is HTTPS. Check this parameter to accept certificate even if certificate authority is unknown.

Ignore Wrong SSL Certificate Usage

This parameter applies only if the URL protocol is HTTPS. Check this parameter to accept malformed certificates.

Ignore SSL Certificate Name Mismatch

This parameter applies only if the URL protocol is HTTPS. Check this parameter to accept a certificate even if the name does not match the visited host.

Ignore Invalid SSL Certificate Date (expired certificate)

This parameter applies only if the URL protocol is HTTPS. Check this parameter to accept a certificate even if it has expired.

Outputs

Bearer Access Token

Parsed bearer token that the responding server provides. Only supports JSON server response.

Exit Paths

Success

This path is taken if the operation is successful and a bearer token is parsed from JSON.

Unknown Host

This path is taken if an invalid or unknown hostname is used (for example, DNS lookup failed).

Timeout

This path is taken if the request timed out.

Connection Failure

This path is taken if a valid HTTP or HTTPS session was not established. For example, a failure to establish a HTTPS connection due to the absence of a client certificate when the Client SSL Certificate parameter is true.

Failure

This path is taken for any failure other than timeout or connection failure.

JSON Builder

This REST tool accepts a list of names, a list of values, and a list of value types to build a JSON object of name/value pairs. This tool builds the name/value pairs and inner JSON alphabetically by name. For more information, refer to the [Interaction Designer REST APIs Developer's Guide](#) in the PureConnect Documentation Library.

Inputs

Names List

The list of names. The length of this list must match the length of the list in the Values List parameter.

Values List

The parallel list of the value for each name in the Names List parameter. The length of this list must match the length of the list in the Names List parameter.

Value Type List

A parallel list of the object type for each output value found in the Values List parameter. The length of this list must match the length of the list in the Names List parameter.

Outputs

JSON string

The variable to store the output.

Exit Paths

Success

This path is taken if the operation is successful.

Failure

This path is taken if the operation fails.

List Length Not Equal

This path is taken if the lengths of the input lists are not equal.

Value Type Mismatch

This path is taken if a value cannot be converted to the listed data type.

Invalid Value Type

This path is taken if a data type provided by the Value Type List parameter does not match an accepted data type (for example, string, bool, double, float, int, array, or object).

JSON Parser

This REST tool outputs a list of names, a list of values, and a list of value types from a JSON object. This tool parses the name/value pairs and inner JSON alphabetically by name. For more information, refer to the [Interaction Designer REST APIs Developer's Guide](#) in the PureConnect Documentation Library.

Inputs

JSON string

The JSON object to parse. For this tool to correctly parse the JSON object, all of the name elements in the name/value pairs listed in the JSON object must be unique.

Outputs

Names List

The list of parsed names.

Values List

The parallel list of the parsed value for each name in the Names List parameter.

Value Type List

A parallel list of the data type for each value found in the Values List parameter. Data types can include: string, bool, double, array, object (JSON), and null. These data types correspond to the JSON data types: string, Boolean, number, array, object (JSON), and null. Integer and floating-point output as a double type (corresponding to JSON's number type).

Exit Paths

Success

This path is taken if the operation is successful.

Failure

This path is taken if the operation fails.

JSON Parse Failure

This path is taken if the JSON library cannot parse the JSON object in the JSON String parameter. This can result from a malformed or missing JSON object.

REST HTTP Request

This REST tool issues an HTTP REST request to the specified URL. This tool supports GET, POST, PUT, PATCH, and DELETE calls.

This tool:

- Expects the certificate file to exist in a specific directory: I3\IC\Certificates\REST. Create the I3\IC\Certificates\REST directory if it does not exist. The I3\IC\Certificates\REST directory is not created by default.

Use the CertTrustU.exe to install a client certificate. The certificate must be in PEM format. For more information, see [PureConnect Security Features Technical Reference](#) in the PureConnect Documentation Library.

- Expects to receive a URL with special characters already URL encoded. The tool does not automatically URL encode special characters. For example, to URL encode Ñ, use %C3%B1.
- Supports many concurrent requests.
- Does not use Windows Certificate store to check for validity of the certificate and utilizes openssl to do the verification.

To obtain more information on any failure, check logs of Interaction Processor or the response body and headers for failure information.

For more information, refer to the [Interaction Designer REST APIs Developer's Guide](#) in the PureConnect Documentation Library.

Inputs

URL

The URL of the request. You must URL encode any special characters in the URL. For example, to URL encode Ñ, use %C3%B1.

Proxy Uri

(Optional) String URI of the HTTP proxy to use for REST calls. This parameter supports the use of a forward proxy to retrieve content from the origin (API) server. This parameter must include the protocol used. For example: http://env4-revproxy1.ininlab or http://192.168.1.10.

HTTP Method

The HTTP method to execute. HTTP methods include GET, POST, PUT, PATCH, and DELETE.

Additional HTTP Headers

(Optional) Additional HTTP headers to include in the REST call other than content type and authorization with a bearer token.

Bearer Token

(Optional) The bearer token string to add to the HTTP header.

Content Type

The content type to add to the header of the REST call. If this parameter contains an empty string, the default (application/json) is used.

Raw Request Body

The body of the request to send with the REST call.

Request Timeout [s]

Maximum time in seconds to wait for request to be sent and response received before terminating the REST call request. Default: 20 seconds.

Client SSL Certificate

(Optional) The name of a certificate in the \\13\IC\Certificates\REST directory on the PureConnect server to use for Client Authentication. The certificate must be in PEM format and contain the unencrypted private key.

Ignore Unknown SSL Certificate Authority

This parameter applies only if the URL protocol is HTTPS. Check this parameter to accept certificate even if certificate authority is unknown.

Ignore Wrong SSL Certificate Usage

This parameter applies only if the URL protocol is HTTPS. Check this parameter to accept malformed certificates.

Ignore SSL Certificate Name Mismatch

This parameter applies only if the URL protocol is HTTPS. Check this parameter to accept a certificate even if the name does not match the visited host.

Ignore Invalid SSL Certificate Date (expired certificate)

This parameter applies only if the URL protocol is HTTPS. Check this parameter to accept a certificate even if it has expired.

Outputs

Status Code

The API server response status code.

Status Text

The API server response status text.

Response Headers

The API server response headers.

Response Body

The API server response body.

Exit Paths

Success

This path is taken if the operation is successful.

Unknown Host

This path is taken if an invalid or unknown hostname is used (for example, DNS lookup failed).

Timeout

This path is taken if the request timed out.

Connection Failure

This path is taken if a valid HTTP or HTTPS session was not established. For example, a failure to establish a HTTPS connection due to the absence of a client certificate when the Client SSL Certificate parameter is true.

Failure

This path is taken for any failure other than unknown host, timeout, or connection failure.

SMS

Overview of SMS Tools

Short Message Services (SMS) tools allow users to exchange text messages over cell phones. To make use of SMS Support, you will need to connect with an SMS Broker. An SMS Broker is a company that takes care of routing the SMS Messages to cell phones and collects SMS Messages from cell phones.

Click on one of the following tools for more information about that tool:

[SMS Append Destination](#)

[SMS Append Destinations](#)

[SMS Append Message](#)

[SMS Create](#)

[SMS Get Results](#)

[SMS Complete Send](#)

[SMS Send](#)

[SMS Set Message](#)

SMS Append Destination

This SMS tool is used to add one destination to an SMSObject. This allows you to send an SMS message to several destinations at once. Typically, the main or primary destination is filled in the SMS Send tool while any additional destination is added with this tool. This tool needs to be called before using the [SMS Send](#) tool.

Inputs

SMS Identifier

The unique identifier for the SMS Object.

Destination

The telephone number of the intended recipient of the SMS Object.

Exit Paths

Success

This path is taken if the destination is successfully added to the SMS Object.

Failure

This path is taken if the operation fails.

SMS Append Destinations

This SMS tool is used to add multiple destinations to an SMSObject. This allows you to send an SMS message to several destinations at once. Typically, the main or primary destination is filled in the [SMS Send](#) tool while any additional destination is added with this tool. Therefore, this tool should be called before using the [SMS Send](#) tool.

Inputs

SMS Identifier

The unique identifier for the SMS Object.

List of Destinations

The list of telephone numbers of the intended recipients of the SMS Object.

Exit Paths

Success

This path is taken if the destinations is successfully added to the SMS Object.

Failure

This path is taken if the operation fails.

SMS Append Message

This SMS tool appends a new message at the end of the message list.

Note: It is possible for a single SMS object to carry several messages that will be sent to each destination. Depending on the SMS broker, sending these messages can require one or more requests.

Inputs

SMS Identifier

The unique identifier for the SMS object.

Message

The text/binary file to be added to the SMS object.

Exit Paths

Success

This path is taken if the message is successfully appended.

Failure

This path is taken if the operation fails.

SMS Complete Send

This SMS tool requests the SMS Server to actually send the SMS Object to the SMS Broker. The message must have been previously sent to the SMS Server using the [SMS Send](#) tool.

Inputs

SMS Identifier

The unique identifier for the SMS Object.

Exit Paths

Success

This path is taken if the SMS Object is successfully sent to the SMS Broker.

Failure

This path is taken if the operation fails.

SMS Create

This SMS tool allows handlers to create SMS Objects from scratch.

Inputs

Queue Identifier

The queue containing the SMS Object.

Outputs

SMS Identifier

The unique identifier for the SMS Object.

Exit Paths

Success

This path is taken if the SMS Object is successfully created.

Failure

This path is taken if the operation fails.

SMS Get Results

This SMS tool queries the specified SMS Object to retrieve its return code list and its error message list.

Inputs

SMS Identifier

The unique identifier for the SMS Object.

Outputs

Return Codes

The numbers representing the results of the last send operation.

Results

The error strings that correspond to the Return Codes.

Exit Paths

Success

This path is taken if the return code and error message lists were successfully retrieved.

Failure

This path is taken if the operation fails.

SMS Send

This SMS tool allows handlers to send SMS Objects to the SMS Server.

Note: This tool does not actually send the message to the SMS Broker. That is done with the [SMS CompleteSend](#) tool. This tool initiates the handler System_OutgoingSMS which calls the SMS CompleteSend tool.

Inputs

SMS Identifier

The unique identifier for the SMS Object.

Remote Name

The name of the recipient of the SMS Object. If the message is to be sent to more than one recipient, this parameter should be left blank. Multiple Remote Names should be added to the EicRemoteName attribute using the QueueManager's vector functions.

Remote Telephone Number

The telephone number of the intended recipient of the SMS Object. This parameter may contain multiple phone numbers. This allows text/binary messages to be sent to several cell phones.

Message

The text/binary file to be added to the SMS Object.

Binary Message?

Check this box if the message is binary.

Truncate to 160 Characters

Check this box if you want the text of a non-binary message to be truncated to 160 characters.

Priority

The priority of the SMS Object. A value of 0 (default) denotes normal priority, use 1 for High priority, and 2 for Urgent.

Note: High and Urgent priorities often require a special subscription with the SMS Broker. Sending messages at these priorities without the necessary subscription may cause an exception.

SMS Broker Class

This integer corresponds to a class attribute for the SMS Broker. The following values may be used:

- 0: Immediate display. The message will not be registered by the mobile phone.
- 1: The message will be stored on the Mobile Phone.
- 2: The message will be stored on the SIM card.
- 3: The message will be stored on the Terminal Equipment. Used for SIM toolkit and Over The Air short messages.
- 4: None (default)

SMS Broker Notification

If used, there will be a Status Report (SR) message generated when the MT message is sent. SMS Notifications can be sent when the message is delivered, not delivered, pending, etc.

Note: Notifications usually requires a special subscription with the SMS Broker. Asking for a notification without the necessary subscription will either cause an exception or simply get ignored, depending on the SMS Broker. Check with your SMS Broker to confirm the values associated with valid notification types.

Timeout

The number of seconds to wait for the message to be sent. Enter a value of zero to use the system default timeout period.

Number of tries

The number of times to try the operation if it fails.

Codepage

This integer is the EBCDIC code page to use. This field is only available if the EBCDIC custom encoding is selected. Currently, only the code page 290 is supported. This integer is set to 0 if not used.

Delayed until

If you want to delay the sending of this message, enter the date you want to have the message sent.

Valid until

Date on which the SMS message will expire.

Send As

The SMS Broker to send from. This input can indicate the default, a user, or a workgroup. Leave this input blank to indicate the default which uses the SMS Outbound Routing table to choose the broker account to send from. Use the following format to indicate a user or workgroup:

<userId>|<AssociationType>|<Association>|<Broker>|<BrokerAccountId>|<BrokerAccountLocalAddress>

<userId>

The user ID of the user initiating the SMS message send.

<AssociationType>

U (user) or W (workgroup) for broker accounts associated with users or workgroups.

<Association>

The user ID or workgroup name of the user or workgroup that is associated with the desired broker account.

<Broker>

The name of the broker as configured in Interaction Administrator.

<BrokerAccountId>

The name of the broker account ID as configured in Interaction Administrator.

<BrokerAccountLocalAddress>

The broker account local address as configured in Interaction Administrator.

Notes:

- If any of the information in the Send As parameter is incorrect or not configured correctly in Interaction Administrator (for example, if the association between the account and the workgroup/user does not exist), the default is used.
- If any of the strings in any of these values in the Send As parameter contain pipe characters (|) or ampersands (&), those strings must be escaped as shown in the following example:

```
private string EscapePipeDelimitedString(string str)
{
    str = str.Replace("&", "&amp;");
    str = str.Replace("|", "&#124;");
    return str;
}
```

Exit Paths

Success

This path is taken if the message is successfully sent to the SMS Server.

Failure

This path is taken if the operation fails.

SMS Set Message

This SMS tool resets the message list with the given message. All messages that were previously carried by this SMS object are discarded.

Inputs

SMS Identifier

The unique identifier for the SMS object.

Message

The text/binary file to be set to the SMS object.

Exit Paths

Success

This path is taken if the message is successfully set.

Failure

This path is taken if the operation fails.

SOAP

SOAP Abort Request

This SOAP tool aborts the request. If 'Send Unhandled Response' is False, it does not send a response notification, not even an "Unhandled" response when the Request handle goes out of scope. Aborting a request is useful if a SOAP request handler is registered as Monitor handler, for example for wildcard SOAPAction. Multiple handlers may fire at the same time, but only one must send a response notification to the client.

Outputs

SOAP Request

Handle of the SOAP request.

Send Unhandled Response

Checkbox. Default = False.

Exit Paths

Success

This path is taken if the request is aborted.

Failure

This path is taken if the operation fails.

SOAP Add Body Element

This SOAP tool adds an entry to the body of the SOAP envelope. Use the XML tools on the returned 'Element' node to add rich contents to the element (not just a string).

Note: If the 'Name' argument has no namespace prefix and a 'Namespace' different than "" (default namespace) is specified, a prefix will be synthesized, unless the local name starts with a ':' (which is illegal in XML, and thus signals to this tool *not* to add a synthesized namespace prefix). Adding a prefix can greatly reduce the size of the message if child elements are in no namespace, as otherwise each child element would get an `xmlns=""` attribute.

Inputs

Envelope

Envelope node of the SOAP payload. Can be a document node whose document element is <SOAP-ENV:Envelope> or the node is the element itself.

Name

Fully qualified name of the element to create and add to the body. Please see remarks above for details.

Namespace

Optional string containing the namespace URI of the element. If the parameter is omitted and the name has a namespace prefix, the tool will search in the parent elements for the namespace with the same prefix and make the element a member of this namespace.

Encoding Style

Optional string containing the value of the 'encodingStyle' attribute. Attribute is omitted if not specified or "NONE." Specify "STANDARD" for standard namespace ("http://schemas.xmlsoap.org/soap/encoding/").

Value

Optional string value to set as content of the element.

Replace Existing Body Element

Checking this box will replace the first element in the body that has the same (local) name and namespace. If the body contains multiple elements with the same name and namespace, the remaining ones are not modified.

By default, this box is **not** checked and the element is added as the last child of the body.

Delete All Existing Body Elements

If this box is checked, all existing elements will be removed from the body prior to adding the new element. By default, this box is **not** checked and the new element is appended to the child list of the body.

Outputs

Body Element

Node of the element that has just been added.

Exit Paths

Success

This path is taken if the element is successfully added.

Failure

This path is taken if the operation fails.

SOAP Add Header Element

This SOAP tool creates a header element and adds it to the given envelope. If the envelope doesn't yet have a header, one will be inserted before the Body element.

Note: If the 'Name' argument has no namespace prefix and a 'Namespace' different than "" (default namespace) is specified, a prefix will be synthesized, unless the local name starts with a ':' (which is illegal in XML, and thus signals to this tool *not* to add a synthesized namespace prefix). Adding a prefix can greatly reduce the size of the message if child elements are in no namespace, as otherwise each child element would get an `xmlns=""` attribute.

Inputs

Envelope

Envelope node of the SOAP payload. Can be a document node whose document element is `<SOAP-ENV:Envelope>` or the node is the element itself.

Name

Fully qualified name of the header element to create and add to the header.

Namespace

Optional string denoting the namespace URI of the element. If the parameter is omitted and the name has a namespace prefix, the tool will search in the parent elements for the namespace with the same prefix and make the element a member of this namespace.

Must Understand

This Boolean sets the value of the 'mustUnderstand' attribute:

False mustUnderstand="0"

True mustUnderstand="1"

Not specified: No 'mustUnderstand' attribute is added.

Actor URI

String Optional setting the value of the 'actor' attribute.

Encoding Style

Optional string setting the value of the 'encodingStyle' attribute. Attribute is omitted if not specified or "NONE". Specify "STANDARD" for standard namespace ("http://schemas.xmlsoap.org/soap/encoding/").

Value

Optional string value to set as content of the element.

Replace Existing Header Element

If this box is checked, this tool will replace first element in the body that has the same (local) name and namespace. If the body contains multiple elements with the same name and namespace, the remaining ones are not modified. By default, this box is **not** checked and the element is added as the last child of the body.

Delete All Existing Header Elements

Check this box to remove all existing elements from the body prior to adding the new element. By default, this box is **not** checked and the element is appended to the child list of the body.

Outputs

Header Element

Node of the element that has just been inserted.

Exit Paths

Success

This path is taken if the header element is successfully added.

Failure

This path is taken if the operation fails.

SOAP Add RPC Parameter

This SOAP tool is a convenience tool for composing RPC requests or responses. It adds a parameter element to the first element in the body of the envelope, which represents the method in RPC requests. Use the XML tools to add complex data (not just a string) to the parameter by manipulating the returned 'Parameter Element' node.

Note: The <Body> element must have a child element (method element). Otherwise this tool fails. When using 'SOAP Create Envelope', you must add a method element using 'SOAP Add Body Element'. The 'SOAP Create RPC Response' tool already adds a method element.

Inputs

Envelope

Envelope node of the SOAP payload. Can be a document node whose document element is <SOAP-ENV:Envelope> or the node is the element itself.

Name

String denoting the qualified name of the parameter.

Namespace

Optional string containing the Namespace URI of the element. If the parameter is omitted and the name has a namespace prefix, the tool will search in the parent elements for the namespace with the same prefix and make the element a member of that namespace.

Value

Optional value of the parameter.

Outputs

Parameter Element

Node of the element that just has been added to the method element.

Exit Paths

Success

This path is taken if the parameter element is successfully added.

Failure

This path is taken if the operation fails.

SOAP Base64 Decode

This SOAP tool decodes the base64 encoded string into the binary representation and converts it to UNICODE based on the specified character set. Thus, the character set argument specifies the character set of the base-64 encoded data.

Inputs

Encoded Data

Base64 encoded data.

Character Set

Character set of the base64 encoded data. Optional. Default: 'UTF-8.'

Outputs

Decoded Data

Data after decoding from Base64 and transforming from 'Character Set' to UNICODE.

Exit Paths

Success

This path is taken if the encoded string is successfully decoded.

Failure

This path is taken if the operation fails.

SOAP Base64 Decode To File

This SOAP tool decodes the base64 encoded string into the binary representation and writes the data to the specified file as binary data.

Inputs

Encoded Data

Base64 encoded data

Filename

Filename and path of the file to which to write the decoded data.

Append To Existing File

Check this box to create a new file or append to an existing file. Leave this box cleared to create a new file or truncate an existing file.

Exit Paths

Success

This path is taken if the file is successfully decoded.

Access Denied

This path is taken if access to the specified file is denied.

Failure

This path is taken if the operation fails.

SOAP Base64 Encode

This SOAP tool converts the string (which is UNICODE) into the specified character set (default = UTF-8) and encodes the resulting data into a Base64 string. Characters that cannot be translated to the destination character set will be represented as '?'. Wide character sets, such as UTF-16 are currently not supported.

Note: SOAP does not mandate a maximum line width for base64 encoded data. Some other protocols, such as mime, do.

Inputs

Data

String to encode Base64

Character Set

Character set to convert data into before encoding. Optional. Default: 'UTF-8'

Max Line Width

Maximum width of a line in characters. The default value of -1 is interpreted as an unlimited line width.

Line Separator

String inserted as line separator. Default: "\r\n" (CR/LF)

Outputs

Encoded Data

String after encoding data Base64.

Exit Paths

Success

This path is taken if the data is successfully encoded.

Failure

This path is taken if the operation fails.

SOAP Base64 Encode File

This SOAP tool reads the specified file as binary data and encodes it into a base64 string.

Note: SOAP does not mandate a maximum line width for base64 encoded data. Some other protocols, such as mime, do.

Inputs

Filename

File name and path of the file to encode.

Max Line Width

Maximum width of a line in characters. The default value of -1 is interpreted as an unlimited line width.

Line Separator

String inserted as line separator. Default: "\r\n" (CR/LF)

Outputs

Encoded Data

Base64 encoded content of the file

Exit Paths

Success

This path is taken if the file is successfully decoded.

File Not Found

This path is taken if the specified file cannot be found.

Access Denied

This path is taken if access to the specified file is denied.

Failure

This path is taken if the operation fails.

SOAP Create Array

This SOAP tool turns an element, for example an RPC parameter, into a SOAP array. The array is created for values supplied as list of strings or just a number of empty elements that can be populated with complex data. The following is a sample array as produced by this tool (default argument):

```
<Element xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
SOAP-ENC:arrayType="xsd:string[5]"
xsi:type="SOAP-ENC:Array">
<xsd:string>first</xsd:string>
<xsd:string>second</xsd:string>
```

```
<xsd:string>third</xsd:string>
<xsd:string>fourth</xsd:string>
<xsd:string>fifth</xsd:string>
</Element>
```

Notes: If the element already has child elements, they are all removed before the array elements are added.

The array items may be user defined (complex) types. Use the 'XML Get Next Item' tool to iterate through the 'Item Elements' collection and populate the items. For example:

```
<Element xmlns:ns1="uri:my-order-type"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
SOAP-ENC:arrayType="ns1:Order[3]"
xsi:type="SOAP-ENC:Array">
<ns1:Order>
<ns1:Product>Watchmacallit</ns1:Product>
<ns1:Quantity>3</ns1:Quantity>
<ns1:Price>19.99</ns1:Price>
</ns1:Order>
<ns1:Order>
<ns1:Product>Doodleany</ns1:Product>
<ns1:Quantity>9</ns1:Quantity>
<ns1:Price>12.49</ns1:Price>
</ns1:Order>
<ns1:Order>
<ns1:Product>Ozadingdong</ns1:Product>
<ns1:Quantity>1</ns1:Quantity>
<ns1:Price>43.15</ns1:Price>
</ns1:Order>
</Element>
```

Please refer to <http://www.w3.org/TR/xmlschema-0> or <http://www.w3.org/TR/xmlschema-2> for details on the XML Schema Datatypes.

Inputs

Element

Node of the parameter to turn into an array.

Values

List of strings to set as the array items. If not specified, empty elements will be created.

Size

Size of the array. If not specified, the length of the 'Values' list specifies the size. If both a 'Values' and 'Size' argument are given, the 'Size' has precedent and either not all items of the 'Values' list are included or the array is padded with elements containing the 'Default Value'.

Default Value

Optional. Default array item value for padding items (if 'Size' is larger than size of 'Values' or no 'Values' defined).

Default: No value (padding elements will be empty)

Array Type

Optional. Type of the array. The argument may either be just the type name or have schema namespace prefix, such as `xsd:string`. If the type argument does not have a prefix, `xsd` will be used.

Default: `xsd:string`

Type Namespace

Optional. Namespace of the array type.

Default: <http://www.w3.org/2001/XMLSchema>

Encoding Prefix

Optional. Prefix of the encoding namespace (<http://schemas.xmlsoap.org/soap/encoding/>).

Default: SOAP-ENC

Item Element Name

Optional. Qualified name of the array items.

Default: Qualified array type (thus, the default item element name is `xsd:string`).

Item Element Namespace

Optional. Namespace of the array items.

Default: Namespace of the prefix of 'Item Element Name'. If no prefix, empty namespace.

XSI Namespace

Optional. XML Schema Instance namespace.

Default: <http://www.w3.org/2001/XMLSchema-instance>

XSI Namespace Prefix

Optional. Prefix of the schema instance namespace.

Default: xsi

Include XSI Type Declaration

This box is checked by default and will add XSI type declaration for the SOAP Array. If all parameters are default, the declaration is: `xsi:type="SOAP-ENC:Array"`.

Uncheck this box if you do not want to add a type declaration for the array.

Declare Namespaces in Envelope

This box is checked by default and will declare the namespaces in the Envelope element (if they aren't already). If any of the parent elements already has a NS declaration for a prefix and the namespace URI is different, the declaration will be added to the element, and not the Envelope.

If unchecked, the namespaces will be declared in the element itself.

Return Item Element Collection

This box is checked by default and returns collection of 'Array Items' containing all items of the array.

If unchecked, collection of array items will not be returned. I.e., 'Array Items' is returned as NULL.

Outputs

Item Elements

Node Iterator pointing to first element of a collection containing the nodes of the array items.

Count

Number of items in the array.

Note: this value is returned, even if 'Return Item Collection' is False.

Exit Paths

Success

This path is taken if the array is successfully created.

Empty

This path is taken if the designated node is empty.

Failure

This path is taken if the operation fails.

SOAP Create Envelope

This SOAP tool creates a new SOAP envelope. To simplify composing RPC requests, where the first child element of the <Body> element is the method to invoke, the 'RPC Method Name' and 'RPC Method Namespace' argument can be used as shortcut. The same can be achieved by invoking 'SOAP Add Body Element' after creating the envelope.

Thus, this tool creates the following XML document:

```
<?xml version="1.0" encoding="{XML Encoding}" ?>
<{Envelope Prefix}:Envelope
xmlns:{Envelope Prefix}="http://schemas.xmlsoap.org/soap/envelope/"
{Envelope Prefix}:encodingStyle="{Encoding Style}"
{Declare Namespaces}>
<{Envelope Prefix}:Body>
[<{RPC Method Name}></{RPC Method Name}>]
</{Envelope Prefix}:Body>
</{Envelope Prefix}:Envelope>
```

The 'Declare Namespaces' argument is used to declare namespaces in the envelope that will be used in other elements, such as the **xsd** or **xsi** prefixes for typed arguments. It keeps the size of the envelope low, as otherwise each element that uses a prefix will contain **xmlns** attributes.

Note: If the 'RPC Method Name' argument has no namespace prefix and an 'RPC Method Namespace' different than "" (default namespace) is specified, a prefix will be synthesized, unless the local name starts with a ':' (which is illegal in XML, but signals to this tool *not* to add a synthesized namespace prefix). Adding a prefix can greatly reduce the size of the message if child elements are in no namespace (usually parameters are in the default namespace), as otherwise each child element would get a **xmlns=""** attribute.

Inputs

XML Encoding

Optional string character encoding to be used for the XML document. If omitted, "UTF-8" is used. See above.

Envelope Prefix

Namespace prefix for the envelope namespace. If not specified, the default "SOAP-ENV" is used.

Encoding Style

Space separated list of namespaces specifying the encoding style (value of the 'encodingStyle' attribute). If not specified or "STANDARD" is passed as string, "http://schemas.xmlsoap.org/soap/encoding/" is used.

The encodingStyle attribute is omitted if "NONE" is specified.

RPC Method Name

Fully qualified name of the method element (first child element of the body element).

If not specified, no method element will be added.

RPC Method Namespace

Namespace of the method element. Optional.

Declare Namespaces

Space delimited list of namespace declarations of the form `xmlns:{prefix}={URI}` to be declared in the envelope. See remarks.

Selection Namespaces

String Optional. Space delimited list of namespace declarations to be set as selection namespaces for the XPath queries. If argument not specified, the envelope prefix and the 'Declare Namespace' namespaces will be set as selection namespaces.

Note: Mapping for envelope prefix will always be added.

Outputs

Envelope

XML document node with Envelope as document element.

Exit Paths

Success

This path is taken if the envelope is successfully created.

Failure

This path is taken if the operation fails.

SOAP Create Fault Response

This SOAP tool copies the request envelope and replaces all children of the <Body> element with a single <Fault> element. It thus combines the [SOAP Create Envelope](#) and [SOAP Set Fault](#) tools.

Note: The selection namespaces from the source envelope document are copied to the response envelope document as well.

Inputs

Envelope

Envelope node of the request SOAP payload. Can be a document node whose document element is <SOAP-ENV:Envelope> or the node is the element itself.

Fault Code

String to set as value of the <faultcode> element. String must not be empty.

Fault String

String to set as value of the <faultstring> element. Should be set to provide human readable information.

Fault Actor

String to set as value of the <faultactor> element. If argument is not specified, no <faultactor> element is added.

Create Detail Element

Leave this box checked if you want to create an empty <detail> element. Clear this box if you do not want to have a <detail> element created.

Note: According to the SOAP spec, a <detail> element must be present if the fault is because the <Body> could not be processed successfully.

Copy Header

Leave this box checked to copies the <Header> element and its contents from the source envelope. Clear this box if you do not want to copy the <Header> element from the source envelope.

Outputs

Response Envelope

Document node of the response envelope.

Detail Element

Node of the <detail> element of the <Fault> element. If 'Create Detail Element' is False, a NULL node is returned.

Exit Paths

Success

This path is taken if the fault response is successfully created.

Failure

This path is taken if the operation fails.

SOAP Create RPC Response

This SOAP tool is a convenience tool for composing the response envelope for an RPC request. It copies the source envelope and replaces the method element in the body with an element that has the same name but "Response" added to its name. It also adds a <Result> element as child of the method element.

Usually, the type of the return value is given by the service description and doesn't need to be included in the <Result> element. However, the service may define the type as **xsd:anyType**, for example for VARIANT types. In this case, the type must be included in the argument. The 'Return Value Type' argument permits specifying the type of the result value. For example, if a type of "double" is specified, the <Result> element will look as follows:

```
<Result xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xsd:double">1234.567</Result>
```

Notes: The selection namespaces from the source envelope document are copied to the response envelope document as well.

The tool fails if the request body does not contain a method element.

Inputs

Envelope

Envelope node of the request SOAP payload. Can be a document node whose document element is <SOAP-ENV:Envelope> or the node is the element itself.

Method Name Mask

Optional mask to create the name of the response method.

The string passed here may contain the following substitution tags:

%1 Namespace prefix of the first child element of the <Body> element (RPC method).

%2 Base name of the first child element of the <Body> element (RPC method).

%{ Treat everything up to closing **}** as XPath query to be run against the 'Envelope' node and substitute the value of the first node found into element name string.

%% '%' character

Default: "%1:%2Response".

Method Namespace

Namespace of the method element. If not specified, namespace of request method is used.

Result Element Name

Name of the return value element (first child of the method element).

Default: "result"

Result Element Namespace

Namespace URI of the result element. If the parameter is omitted and the name has a namespace prefix, the tool will search in the parent elements for the namespace with the same prefix and make the element a member of that namespace.

Return Value

String containing the return value of the method. This parameter does not parse XML data. This toolstep uses the return value as content of the <Result> child element. Optional.

No Return Value (void response)

Check this box if you do not want to add a <Result> element. Leave this box unchecked if you do want to add a <Result> element.

Copy Header

Check this box to copy the <Header> element and its contents from the source envelope. Leave this box unchecked if you do not want to copy the <Header> element from the source envelope.

Copy Method Element Attributes

Check this box to copy all attributes of the request method element into response method element. Leave this box unchecked if you do not want to copy the attributes from request method element.

Outputs

Response Envelope

Document node of the response envelope.

Method Element

Node of the response method element.

Result Element

Node of the <Result> element in the method element.

Exit Paths

Success

This path is taken if the RPC response is successfully created.

Failure

This path is taken if the operation fails.

SOAP Expects Response

This SOAP tool takes a different exit path depending on whether the SOAP request requires a response (YES) or not (NO). If the request expects a response and the handler exits (the SOAP Request handle goes out of scope) without having invoked 'SOAP Send Response', a Response Notification is sent back with the 'Unhandled' flag set to true.

Inputs

SOAP Request

Handle of the SOAP request.

Exit Paths

Yes

This path is taken if the SOAP request requires a response.

No

This path is taken if the SOAP request does not require a response.

SOAP Get Body

This SOAP tool retrieves the Body element from the SOAP envelope. A body must exist and if it can't be found, the tool exits through 'Failure' and attaches error information to the envelope.

Inputs

Envelope

Envelope node of the SOAP payload. Can be a document node whose document element is <SOAP-ENV:Envelope> or the node is the element itself.

Outputs

Body

Node of the <SOAP-ENV:Body> element.

Exit Paths

Success

This path is taken if the Body element is retrieved.

Failure

This path is taken if the SOAP envelope does not contain a Body element.

SOAP Get Body Element

This SOAP tool retrieves the first body element that matches the given base name and namespace. If no namespace is specified, the first element matching 'Base Name' is returned. Returns the first element in the body if neither a name nor namespace is given.

Inputs

Envelope

Envelope node of the SOAP payload. Can be a document node whose document element is <SOAP-ENV:Envelope> or the node is the element itself.

Base Name

Base name of the element to return. If no name given, the first entry in the body in 'Namespace' is returned. This corresponds to the element of the method for RPC requests.

Namespace

Namespace of the element to return. Optional.

Retrieve Value

Check this box to return the node value. Leave this box unchecked if you do not want to retrieve the node value.

Outputs

Body Element

Child element of the <SOAP-ENV:Body> element that has the given base name and namespace. NULL node if the element is not in the body.

Element Base Name

Base name of the returned element.

Element Namespace

Namespace URI of the returned element.

Value

Value of the body element (if 'Retrieve Value' = True).

Exit Paths

Success

This path is taken if the specified Body element is returned.

Not Found

This path is taken if the specified namespace is not found.

Failure

This path is taken if the operation fails.

SOAP Get Fault

This SOAP tool retrieves fault information from the SOAP envelope. If there is no <Fault> element in the envelope, the 'No Fault' exit is taken and NULL elements and empty strings are returned.

Note: If the envelope is read-only, the returned elements will be read-only too.

Inputs

Envelope

Envelope node of the SOAP payload. Can be a document node whose document element is <SOAP-ENV:Envelope> or the node is the element itself.

Outputs

Fault Element

Node of the <Fault> element.

Fault Code

Value of the <faultcode> element. It provides programmatic information about the fault.

Fault String

Value of the <faultstring> element. It provides human readable information about the fault.

Fault Actor

Value of the <faultactor> element. It provides the URI of the source of the fault.

Detail Element

Node of the <detail> element. It is used to transfer application specific fault information. NULL Node if there is no <detail> element.

Exit Paths

Success

This path is taken if

No Fault

This path is taken if there is not <Fault> element in the envelope.

Failure

This path is taken if the operation fails.

SOAP Get Header

This SOAP tool retrieves the header element from the SOAP envelope if it has one.

Inputs

Envelope

Envelope node of the SOAP payload. Can be a document node whose document element is <SOAP-ENV:Envelope> or the node is the element itself.

Outputs

Header

Node of the <SOAP-ENV:Header> element. NULL node if the envelope contains no header.

Exit Paths

Success

This path is taken if the header is successfully retrieved.

No Header

This path is taken if the SOAP envelope does not contain a Header element.

Failure

This path is taken if the operation fails.

SOAP Get Header Element

This SOAP tool retrieves the first header element that matches the given base name and namespace. Returns the first element in the header if neither a name nor namespace is given.

Inputs

Envelope

Envelope node of the SOAP payload. Can be a document node whose document element is <SOAP-ENV:Envelope> or the node is the element itself.

Base Name

Base Name of the element to return. Optional.

Namespace

Namespace of the entry to return. Optional.

Retrieve Value

Leave this box checked to return the node value. Clear the box if you do not want to retrieve the value.

Outputs

Header Element

Child element of the <SOAP-ENV:Header> element that has the given base name and namespace. NULL node if the envelope contains no header or the element is not in the header.

Element Base Name

Base name of the returned element.

Element Namespace

Namespace URI of the returned element.

Value

Value of the element (if 'Retrieve Value' = True).

Exit Paths

Success

This path is taken if the header element is retrieved.

Not Found

This path is taken if the element can't be found.

No Header

This path is taken if the envelope doesn't have a header.

Failure

This path is taken if the operation fails.

SOAP Get Header Elements

This SOAP tool returns iterator to a list of header elements filtered by the given arguments. Takes the 'None' exit if envelope has no header or none of the header elements matched the filter criteria.

Inputs

Envelope

Envelope node of the SOAP payload. Can be a document node whose document element is <SOAP-ENV:Envelope> or the node is the element itself.

Base Name

Only include elements with this base name. Optional.

Namespace

Only include elements in this namespace. Optional.

Must Understand

If left unspecified, entries will not be filtered on 'mustUnderstand'. Otherwise, this field has the following possible values:

False Return header entries whose 'mustUnderstand' attribute is "0" (or no attribute is specified)

True Return header entries whose 'mustUnderstand' attribute is "1".

Actor URIs

Space separated list of actor URIs. Only elements whose actor attribute has one of these namespaces is returned. If not specified, don't filter on actor namespace.

Outputs

Header Elements

Iterator to collection of header entries. Use the 'XML Get Next Node' tool to iterate over collection.

Count

Number of items in the Header Entries collection.

Exit Paths

Success

This path is taken if the header element is retrieved.

None

This path is taken if the element can't be found.

No Header

This path is taken if the envelope doesn't have a header.

Failure

This path is taken if the operation fails.

SOAP Get Next RPC Parameter

This SOAP tool returns the element node at the current iterator position and returns an iterator to the next position. As the iterator is just a variable, you can make copies at any time to remember a certain position, for example the start position. By using the same variable as input and output iterator, you can easily iterate over the list by connecting the Success path back to this tool (after processing the node, of course).

Note: The tool will fail (take the Failure exit) if the node to which 'Parameter Iterator' points is not an element! This cannot happen if the iterator was obtained through 'SOAP Get RPC Method Info'.

Inputs

Parameter Iterator

Iterator to collection of parameter of a method.

Retrieve Value

Leave this box checked to return the node value. Clear this box if you do not want to retrieve the value. Disable retrieval of value if value is not used and parameter may contain a large XML document.

Outputs

Next Parameter

Iterator pointing to next parameter in the list.

Parameter Element

Node of the parameter element.

Parameter Base Name

Base name of the parameter element.

Parameter Namespace

Namespace URI of the parameter element.

Value

Value of the parameter.

Exit Paths

Success

This path is taken if the element node is returned.

End

The tool takes the 'End' exit when the iterator points to an empty list or the iteration is complete (list traversed to end).

Failure

This path is taken if the operation fails.

SOAP Get Request Info

This SOAP tool queries information from the request handle.

Inputs

SOAP Request

Handle of the SOAP request.

Outputs

Initiator Event

String denoting the notification event that caused the initiator to trigger.

SOAP Action

String denoting the SOAP action code of that request.

Client ID

Client ID (Notifier object id). Integer.

Client Name

String containing the name of the client

Request ID

Integer Request ID (for debugging/tracing purposes).

Payload Size

Size of the request payload in bytes. Integer.

Transport Info Size

Size of the transport information in size. Integer.

Exit Paths

Success

This path is taken if the information is successfully retrieved.

Failure

This path is taken if the operation fails.

SOAP Get RPC Method Info

This SOAP tool is a convenience tool for cracking RPC requests. It retrieves the first child element of the SOAP <Body> element (Method element in RPC requests). It also returns a collection containing the child elements of the method, which constitute the method arguments.

Inputs

Envelope

Envelope node of the SOAP payload. Can be a document node whose document element is <SOAP-ENV:Envelope> or the node is the element itself.

Outputs

Method Element

Node of the method element (first child of the Body).

Method Base Name

Base name of the method element.

Method Namespace

Namespace URI of the method element.

Parameters

Iterator to collection of RPC parameter elements. Use the 'SOAP Get Next RPC Parameter' or 'XML Get Next Node' tool to iterate over collection.

Parameter Count

Number of items in the Parameters collection.

Exit Paths

Success

This path is taken if the first child element is returned.

Fault

This path is taken if the body contains a <Fault> element.

No Method

This path is taken if the body does not contain an element.

Failure

This path is taken if the operation fails.

SOAP Get RPC Parameter

This SOAP tool is a convenience tool for cracking RPC requests. It retrieves a parameter element (child) from the first element in the <Body> element (method in an RPC request).

It returns the first element that matches all of the specified arguments. If 'Base Name', 'Namespace', and 'Index' are undefined, the first element will be returned.

Example: To retrieve the 2nd parameter from the 'Add' method in the calculator example presented in the Introduction, you would specify "Parameter2" as name and "" as namespace, or '1' as index.

Inputs

Envelope

Envelope node of the SOAP payload. Can be a document node whose document element is <SOAP-ENV:Envelope> or the node is the element itself.

Base Name

Base name of the parameter. Optional.

Namespace

Namespace of the parameter. Optional.

Index

Zero based index into parameters of the method. If this parameter is specified, 'Name' and 'Namespace' may be omitted, but if present must match the name and namespace of the parameter.

Retrieve Value

Leave this box checked to return the node value. Clear this box if you do not want to retrieve the value. Disable retrieval of value if parameter contains a large XML document and the value is not used (performance option).

Outputs

Parameter Element

Parameter element.

Parameter Base Name

Base name of the parameter element.

Parameter Namespace

Namespace URI of the parameter element.

Parameter Index

Zero based index of the parameter element in the child list of the method element.

Value

Value of the parameter.

Exit Paths

Success

This path is taken if the element is successfully retrieved.

Not Found

This path is taken if the element can't be found.

Failure

This path is taken if the operation fails.

SOAP Get Transport Info

This SOAP tool returns an XML document containing transport specific (header) data. It allows the client to include any kind of out-of-band data in the request. For example, for HTTP requests, this document contains the HTTP method and a list of the header elements. See appendix for Schema. If there is no transport information data, an empty document is returned and the tool takes the 'No Info' exit. If there is an error (Failure), an empty document is returned which can be queried with 'XML Get Error Info'.

Note: The data may be parsed every time the tool is invoked or cached. This may depend on the specified selection namespaces. The returned document is read-only.

Inputs

SOAP Request

Handle of the SOAP request.

Selection Namespaces

Optional. Space delimited list of namespace declarations to be set as selection namespaces the XPath queries.

Preserve Whitespace

If this box is checked, nonessential whitespaces will be preserved. By default, this box is **not** checked and whitespaces are ignored when parsing the payload.

Validate On Parse

If this box is checked, the tool will validate against the schema during parse. By default, this box is **not** checked and it only verifies for well-formedness.

Resolve Externals

If this box is checked, all resolvable externals (namespaces, DTDs, entity references etc.) will be resolved at parse time. By default, this box is **not** checked and externals are not resolved.

Outputs

Transport Info

Read-only node. XML document containing transport-specific out-of-band information. Empty document if no transport information. See remarks.

Exit Paths

Success

This path is taken if the info is successfully retrieved.

No Info

This path is taken if there is no information to retrieve.

Failure

This path is taken if the operation fails.

SOAP HTTP Request

This SOAP tool issues an HTTP request to the specified URL with the SOAP request envelope as payload. The response body is parsed and returned as response envelope.

The URL may have the following format (also see RFC2396):

```
['http://'] <host> [':' <port>] [ '/' <path> ['?' <query>]]
```

The (UNICODE) string passed as URL is converted to UTF-8 and invalid characters in the resulting string are escaped according to RFC2396 (%<hexvalue>).

The structure of the request sent to the host will be as follows:

```
'POST ' <path> ' HTTP/1.1' CRLF
```

```
'Host: ' <host> [':' <port>] CRLF
'Content-Type: text/xml; charset="' <charset> '"' CRLF
'Content-Length: ' <bodysize> CRLF
'SOAPAction: "' <SOAPAction> '"' CRLF
[<additional headers>]
CRLF
<SOAP envelope body>
```

The 'Additional HTTP Headers' parameter can be used to supply additional HTTP header elements. The headers must have the form `{<name>:' <value> [CR] LF}`*The header elements in this argument have precedence over the default headers generated by the tool. Thus, if the 'Additional HTTP Headers' parameter contains a 'Content-Length' header, it will be used (with potentially unexpected results, of course).

The response body will be parsed and returned as 'Response Envelope' if the content type is text/xml. Otherwise, the body is returned in 'Raw Response Body' and an empty document node is returned as 'Response Envelope'. This document node can be queried for information about what went wrong.

This tool maintains a global cache of the most recently resolved and successfully connected host addresses to improve performance. Each address is kept for at most 5 minutes.

Inputs

Request Envelope

XML Node of the SOAP envelope to send. Can be document node or <Envelope> element node.

URL

URL of the request. See above for details.

SOAP Action

String to be passed as SOAPAction header. The string passed here may contain the following substitution tags:

%1 Namespace of the first child element of the <Body> element (RPC method).

%2 Base name of the first child element of the <Body> element (RPC method).

%{ Treat everything up to closing '}' as XPath query to be run against the 'Request Envelope' node and substitute the value of the first node found into the SOAPAction string.

%% '%' character

If this argument is not specified, the following mask will be used as default: "%1#%2".

The value "NONE" may be specified to suppress addition of the SOAPAction header.

Additional HTTP Headers

Additional HTTP Headers, separated by LF characters (\n). Optional. See above for details.

Selection Namespaces

Selection namespaces to set in response envelope document. Optional.

Default: Copy selection namespaces from request envelope document.

Timeout Value

Maximum time the request may take before timing out (in milliseconds). A value of -1 is interpreted as an infinite timeout period.

Default: 60000 (60 seconds)

Max Response Size

Size limit of the response data. If the data returned by the server exceeds this limit, the data is not processed and the tool fails. This prevents denial of service attacks. Default: 1MB.

Escape URL

Leave this box checked to escape invalid characters in the URL with %<hexvalue> according to RFC2396. Clear this box if the URL is already escaped.

Always Return Raw Response Body

Check this box to return the raw data of the response body as string ('Raw Response Body'). Leave this box cleared if you do not want to return raw response data.

Outputs

Response Envelope

Document node of the response envelope. If an error occurred, an empty document is returned which can be queried using 'XML Get Error Info'.

Status Code

HTTP status code of the response (e.g. 200, 500, etc).

Status Text

HTTP status text of the response (e.g. "OK", "Internal Server Error", etc).

Response Headers

HTTP Headers returned by the server, separated by a LF (\n).

Raw Response Body

Raw data of the response body (data that is parsed as response envelope). This string is only returned if the 'Always Return Raw Response Body' parameter is True, an error occurs, or the response content type is not XML.

Exit Paths

Success

This path is taken if the response contains valid SOAP envelope element(s) and the body is valid XML.

SOAP Fault

This path is taken if the response body contains a <Fault> element.

EmptyResponse

This path is taken if the response body was empty and the HTTP status code was 2xx. Some servers use this to signal success for methods with no result (void).

Unknown Host

This path is taken if an invalid or unknown hostname is used (DNS lookup failed).

Connection Error

This path is taken if it is unable to establish connection to server: connection failed or existing connection was lost prematurely.

HTTP Error

This path is taken if there was an HTTP error (3xx, 4xx, 5xx) and it was not a SOAP Fault.

Parse Error

This path is taken if there was an error parsing the returned XML payload (status was 200 or 500).

Timeout

This path is taken if the request timed out.

Size Limit

This path is taken if the response data exceeded the size limit.

Failure

This path is taken for any other failure. Use 'XML Get Error Info' on the 'Response Envelope' to obtain more information.

SOAP HTTP Request Ex

This SOAP tool issues an HTTP request to the specified URL with the SOAP request envelope as payload. The response body is parsed and returned as response envelope.

This tool is similar to [SOAP HTTP Request](#) but provides additional options and supports SSL secured connections (https). This tool uses the Microsoft IServerXMLHttpRequest COM component to issue the request.

For requests that do not require the added options provided by this tool, the [SOAP HTTP Request](#) tool should be used as it is more efficient (less overhead).

The response body will be parsed and returned as 'Response Envelope' if the content type is text/xml. Otherwise, the body is returned in 'Raw Response Body' and an empty document node is returned as 'Response Envelope'. This document node can be queried for information about what went wrong.

To obtain more information on any failure other than a 'SOAP Fault', use the 'XML Get Error Info' tool on the response envelope XML document returned by this tool.

Inputs

Request Envelope

The optional XML Node of the SOAP envelope to send. Can be document node or <Envelope> element node. It can also be empty to send an empty request.

URL

URL of the request. See above for details.

HTTP Method

The HTTP method to execute. Default: "POST"

SOAP Action

The string to be passed as the SOAPAction header. The string passed here may contain the following substitution tags:

%1 Namespace of the first child element of the <Body> element (RPC method).

%2 Base name of the first child element of the <Body> element (RPC method).

%{ Treat everything up to closing **}** as XPath query to be run against the 'Request Envelope' node and substitute the value of the first node found into the SOAPAction string.

%% '%' character

Default: "%1#%2".

Note: For .NET webservices, use "%1/%2"

The value "NONE" may be specified to suppress addition of the SOAPAction header.

Additional HTTP Headers

Optional. Additional HTTP Headers, separated by LF characters (\n). See remarks for details.

Selection Namespaces

Optional. Selection namespaces to set in the response envelope document.

Default: Copy selection namespaces from the request envelope document.

Username (auth.sites)

Optional. Username for sites that require HTTP authentication. Default: No authentication.

Password (auth. sites)

Optional. Password for sites that require HTTP authentication. Default: No authentication.

DNS Resolve Timeout [s]

Maximum time in seconds to wait for name resolution to complete. Default: 30s

Connect Timeout [s]

Maximum time in seconds to wait for connection to be established. Default: 30s

Request Timeout [s]

Maximum time in seconds to wait for request to be sent and response to be completely received.

Default: 120s (2min)

Max Response Size (bytes)

Optional. Size limit of the response data. If the data returned by the server exceeds this limit, the data is not processed and the tool fails. This prevents denial of service attacks. Default: 1MB.

Escape URL

Unchecked URL is already escaped.

Checked by default. Escape invalid characters in the URL with %<hexvalue> according to RFC2396.

Always Return Raw Response Body

Unchecked by default. Do not return raw response body.

Check to return the raw data of the response body as string ('Raw Response Body').

Client SSL Certificate

Optional. Name of the client certificate to use from the local store. Default or empty string: Pick first certificate in store.

Ignore Unknown SSL Certificate Authority

Unchecked by default. Fail if SSL certificate authority is unknown.

Check to accept SSL certificate even if certificate authority is unknown.

Ignore Wrong SSL Certificate Usage

Unchecked by default. Fail if SSL certificate is malformed (e.g. no subject name).

Check to accept malformed SSL certificates.

Ignore SSL Certificate Name Mismatch

Unchecked by default. Fail if the hostname of the visited host and the server certificate do not match.

Check to accept an SSL certificate even if the certificate name doesn't match the visited host.

Ignore Invalid SSL Certificate Date (expired certificate)

Unchecked by default. Fail if SSL certificate has expired or its date is invalid.

Check to accept an SSL certificate even if it has expired.

Outputs

Response Envelope

Document node of the response envelope. If an error occurred, an empty document is returned which can be queried using 'XML Get Error Info'.

Status Code

HTTP status code of the response (e.g. 200, 500, etc).

Status Text

HTTP status text of the response (e.g. "OK", "Internal Server Error", etc).

Final URL

URL that was actually used to make the request after any redirections. This parameter can be used to check whether a redirect occurred and what the URL of the final server was that served the request.

Response Headers

HTTP Headers returned by the server, separated by a LF (\n).

Raw Response Body

Raw data of the response body (data that is parsed as response envelope). This string is only returned if the 'Always Return Raw Response Body' parameter is True, an error occurs, or the response content type is not XML.

Exit Paths

Success

This path is taken if the response contains valid SOAP envelope element(s) and the body is valid XML.

SOAP Fault

This path is taken if the response body contains a <Fault> element.

Empty Response

This path is taken if the response body was empty and the HTTP status code was 2xx. Some servers use this to signal success for methods with no result (void).

Unknown Host

This path is taken if an invalid or unknown hostname is used (DNS lookup failed).

HTTP Error

This path is taken if there was an HTTP error (3xx, 4xx, 5xx) and it was not a SOAP Fault.

Parse Error

This path is taken if there was an error parsing the returned XML payload (status was 200 or 500).

Timeout

This path is taken if the request timed out.

Size Limit

This path is taken if the response data exceeded the size limit.

Failure

This path is taken for any other failure. Use 'XML Get Error Info' on the 'Response Envelope' to obtain more information.

SOAP HTTP Request Ex2

This SOAP tool issues an HTTP request to the specified URL with the SOAP request envelope as payload. The response body is parsed and returned as response envelope.

This tool is similar to [SOAP HTTP Request](#) but provides additional options and supports SSL secured connections (https). This tool uses the Microsoft IServerXMLHttpRequest COM component to issue the request.

For requests that do not require the added options provided by this tool, the [SOAP HTTP Request](#) tool should be used as it is more efficient (less overhead).

The response body will be parsed and returned as 'Response Envelope' if the content type is text/xml. Otherwise, the body is returned in 'Raw Response Body' and an empty document node is returned as 'Response Envelope'. This document node can be queried for information about what went wrong.

To obtain more information on any failure other than a 'SOAP Fault', use the 'XML Get Error Info' tool on the response envelope XML document returned by this tool.

Inputs

Request Envelope

The optional XML Node of the SOAP envelope to send. Can be document node or <Envelope> element node. It can also be empty to send an empty request.

URL

URL of the request. See above for details.

HTTP Method

The HTTP method to execute. Default: "POST"

SOAP Action

The string to be passed as the SOAPAction header. The string passed here may contain the following substitution tags:

%1 Namespace of the first child element of the <Body> element (RPC method).

%2 Base name of the first child element of the <Body> element (RPC method).

%{ Treat everything up to closing '}' as XPath query to be run against the 'Request Envelope' node and substitute the value of the first node found into the SOAPAction string.

%% '%' character

Default: "%1#%2".

Note: For .NET webservices, use "%1/%2"

The value "NONE" may be specified to suppress addition of the SOAPAction header.

Additional HTTP Headers

Optional. Additional HTTP Headers, separated by LF characters (\n). See remarks for details.

Selection Namespaces

Optional. Selection namespaces to set in the response envelope document.

Default: Copy selection namespaces from the request envelope document.

Username (auth.sites)

Optional. Username for sites that require HTTP authentication. Default: No authentication.

Password (auth. sites)

Optional. Password for sites that require HTTP authentication. Default: No authentication.

Password Encrypted?

Indicates whether or not the password is encrypted (true or false).

DNS Resolve Timeout [s]

Maximum time in seconds to wait for name resolution to complete. Default: 30s

Connect Timeout [s]

Maximum time in seconds to wait for connection to be established. Default: 30s

Request Timeout [s]

Maximum time in seconds to wait for request to be sent and response to be completely received.

Default: 120s (2min)

Max Response Size (bytes)

Optional. Size limit of the response data. If the data returned by the server exceeds this limit, the data is not processed and the tool fails. This prevents denial of service attacks. Default: 1MB.

Escape URL

Unchecked URL is already escaped.

Checked by default. Escape invalid characters in the URL with %<hexvalue> according to RFC2396.

Always Return Raw Response Body

Unchecked by default. Do not return raw response body.

Check to return the raw data of the response body as string ('Raw Response Body').

Client SSL Certificate

Optional. Name of the client certificate to use from the local store. Default or empty string: Pick first certificate in store.

Ignore Unknown SSL Certificate Authority

Unchecked by default. Fail if SSL certificate authority is unknown.

Check to accept SSL certificate even if certificate authority is unknown.

Ignore Wrong SSL Certificate Usage

Unchecked by default. Fail if SSL certificate is malformed (e.g. no subject name).

Check to accept malformed SSL certificates.

Ignore SSL Certificate Name Mismatch

Unchecked by default. Fail if the hostname of the visited host and the server certificate do not match.

Check to accept an SSL certificate even if the certificate name doesn't match the visited host.

Ignore Invalid SSL Certificate Date (expired certificate)

Unchecked by default. Fail if SSL certificate has expired or its date is invalid.

Check to accept an SSL certificate even if it has expired.

Outputs

Response Envelope

Document node of the response envelope. If an error occurred, an empty document is returned which can be queried using 'XML Get Error Info'.

Status Code

HTTP status code of the response (e.g. 200, 500, etc).

Status Text

HTTP status text of the response (e.g. "OK", "Internal Server Error", etc).

Final URL

URL that was actually used to make the request after any redirections. This parameter can be used to check whether a redirect occurred and what the URL of the final server was that served the request.

Response Headers

HTTP Headers returned by the server, separated by a LF (\n).

Raw Response Body

Raw data of the response body (data that is parsed as response envelope). This string is only returned if the 'Always Return Raw Response Body' parameter is True, an error occurs, or the response content type is not XML.

Exit Paths

Success

This path is taken if the if the response contains valid SOAP envelope element(s) and the body is valid XML.

SOAP Fault

This path is taken if the response body contains a <Fault> element.

Empty Response

This path is taken if the response body was empty and the HTTP status code was 2xx. Some servers use this to signal success for methods with no result (void).

Unknown Host

This path is taken if an invalid or unknown hostname is used (DNS lookup failed).

HTTP Error

This path is taken if there was an HTTP error (3xx, 4xx, 5xx) and it was not a SOAP Fault.

Parse Error

This path is taken if there was an error parsing the returned XML payload (status was 200 or 500).

Timeout

This path is taken if the request timed out.

Size Limit

This path is taken if the response data exceeded the size limit.

Failure

This path is taken for any other failure. Use 'XML Get Error Info' on the 'Response Envelope' to obtain more information.

SOAP HTTP Request Ex3

This SOAP tool issues an HTTP request to the specified URL with the SOAP request envelope as payload. The response body is parsed and returned as response envelope.

This tool is similar to [SOAP HTTP Request](#) but eliminates the need to use semaphores to limit the number of SOAP requests. It also supports SSL secured connections (https). This tool is similar to the [SOAP HTTP Request Ex2](#) tool, except that:

- This tool does not use MSXML to send SOAP requests.
- This tool expects the certificate file to be present in a specific directory: I3\IC\Certificates\SOAP.
- The certificate must be in PEM format. For more information about converting a certificate to the PEM format, see [PureConnect Security Features Technical Reference](#) in the PureConnect Documentation Library.
- This tool supports many concurrent requests.
- This tool does not fully support NTLM. If you absolutely need to use NTLM, you should use the [SOAP HTTP Request Ex2](#) tool instead.
- This tool does not use Windows Certificate store to check for validity of the certificate, and it uses openssl to do the verification.

Note: The SOAP HTTP Request Ex3 toolstep does not support the use of a HTTP proxy and therefore requires that the CIC server has a direct routed connection to the web server hosting the SOAP endpoint.

The response body will be parsed and returned as 'Response Envelope' if the content type is text/xml. Otherwise, the body is returned in 'Raw Response Body' and an empty document node is returned as 'Response Envelope'. This document node can be queried for information about what went wrong.

To obtain more information on any failure other than a 'SOAP Fault,' use the 'XML Get Error Info' tool on the response envelope XML document returned by this tool.

Inputs

Request Envelope

The optional XML Node of the SOAP envelope to send. Can be document node or <Envelope> element node. It can also be empty to send an empty request.

URL

URL of the request. See above for details.

HTTP Method

The HTTP method to execute. Default: "POST"

SOAP Action

The string to be passed as the SOAPAction header. The string passed here may contain the following substitution tags:

%1 Namespace of the first child element of the <Body> element (RPC method).

%2 Base name of the first child element of the <Body> element (RPC method).

%{ Treat everything up to closing '}' as XPath query to be run against the 'Request Envelope' node and substitute the value of the first node found into the SOAPAction string.

%% '%' character

Default: "%1#%2".

Note: For .NET webservices, use "%1/%2"

The value "NONE" may be specified to suppress addition of the SOAPAction header.

Additional HTTP Headers

Optional. Additional HTTP Headers, separated by LF characters (\n). See remarks for details.

Selection Namespaces

Optional. Selection namespaces to set in the response envelope document.

Default: Copy selection namespaces from the request envelope document.

Username (auth.sites)

Optional. Username for sites that require HTTP authentication. Default: No authentication.

Password (auth. sites)

Optional. Password for sites that require HTTP authentication. Default: No authentication.

Password Encrypted?

Indicates whether or not the password is encrypted (true or false).

DNS Resolve Timeout [s]

Maximum time in seconds to wait for name resolution to complete. Default: 30s

Connect Timeout [s]

Maximum time in seconds to wait for connection to be established. Default: 30s

Request Timeout [s]

Maximum time in seconds to wait for request to be sent and response to be completely received.

Default: 120s (2min)

Max Response Size (bytes)

Optional. Size limit of the response data. If the data returned by the server exceeds this limit, the data is not processed and the tool fails. This prevents denial of service attacks. Default: 1MB.

Escape URL

Unchecked URL is already escaped.

Checked by default. Escape invalid characters in the URL with %<hexvalue> according to RFC2396.

Always Return Raw Response Body

Unchecked by default. Do not return raw response body.

Check to return the raw data of the response body as string ('Raw Response Body').

Client SSL Certificate

Optional. Name of the client certificate to use from the local store. Default or empty string: Pick first certificate in store.

Ignore Unknown SSL Certificate Authority

Unchecked by default. Fail if SSL certificate authority is unknown.

Check to accept SSL certificate even if certificate authority is unknown.

Ignore Wrong SSL Certificate Usage

Unchecked by default. Fail if SSL certificate is malformed (e.g. no subject name).

Check to accept malformed SSL certificates.

Ignore SSL Certificate Name Mismatch

Unchecked by default. Fail if the hostname of the visited host and the server certificate do not match.

Check to accept an SSL certificate even if the certificate name doesn't match the visited host.

Ignore Invalid SSL Certificate Date (expired certificate)

Unchecked by default. Fail if SSL certificate has expired or its date is invalid.

Check to accept an SSL certificate even if it has expired.

Outputs

Response Envelope

Document node of the response envelope. If an error occurred, an empty document is returned which can be queried using 'XML Get Error Info'.

Status Code

HTTP status code of the response (e.g. 200, 500, etc).

Status Text

HTTP status text of the response (e.g. "OK", "Internal Server Error", etc).

Final URL

URL that was actually used to make the request after any redirections. This parameter can be used to check whether a redirect occurred and what the URL of the final server was that served the request.

Response Headers

HTTP Headers returned by the server, separated by a LF (\n).

Raw Response Body

Raw data of the response body (data that is parsed as response envelope). This string is only returned if the 'Always Return Raw Response Body' parameter is True, an error occurs, or the response content type is not XML.

Exit Paths

Success

This path is taken if the response contains valid SOAP envelope element(s) and the body is valid XML.

SOAP Fault

This path is taken if the response body contains a <Fault> element.

Empty Response

This path is taken if the response body was empty and the HTTP status code was 2xx. Some servers use this to signal success for methods with no result (void).

Unknown Host

This path is taken if an invalid or unknown hostname is used (DNS lookup failed).

HTTP Error

This path is taken if there was an HTTP error (3xx, 4xx, 5xx) and it was not a SOAP Fault.

Parse Error

This path is taken if there was an error parsing the returned XML payload (status was 200 or 500).

Timeout

This path is taken if the request timed out.

Size Limit

This path is taken if the response data exceeded the size limit.

Failure

This path is taken for any other failure. Use 'XML Get Error Info' on the 'Response Envelope' to obtain more information.

SOAP Parse Request Payload

This SOAP tool parses the payload of the request into an XML document. If the 'Validate SOAPAction' parameter is True, the tool checks the SOAPAction field of the request against the payload. It uses the following heuristic to match the action code (legend:

<NS> → namespace of the first body element; <MethodName> → Name of the element [method name]:

1. <NS>
2. <NS> [<AnyCharacter>] <MethodName>
3. <MethodName>
4. [<AnyCharacter>] <MethodName>

This will catch actions such as "uri:my-uri#MyMethod", "http://soap.inin.com/e-faq", "MyMethod" etc. An empty SOAPAction matches all methods.

Notes: The payload data is parsed every time this tool is invoked (i.e. it is not cached). The document is furthermore *not* read-only

and thus may be modified as needed, for example to create the response.

The payload envelope node will still be returned, even if the SOAP Action does not match.

Inputs

SOAP Request

Handle of the SOAP request.

Validate SOAP Action

If this box is checked, the SOAPAction header field will be verified against the payload. If this box is not checked, the SOAPAction header will not be verified against the payload. This box is checked by default.

Action Validation Mask

This optional input parameter is not currently used and will be removed in a future service update.

Selection Namespaces

Optional. Space delimited list of namespace declarations to be set as selection namespaces the XPath queries.

Note: The "SOAP-ENV" prefix will be used irrespective of the actual prefix in the payload.

A declaration mapping "SOAP-ENV" to the envelope namespace will always be added to the declarations, *unless* SOAP-ENV is already declared in the argument.

Preserve Whitespace

If this box is checked, nonessential whitespaces will be preserved. By default, this box is **not** checked and whitespaces are ignored when parsing the payload.

Validate On Parse

If this box is checked, the tool will validate against the schema during parse. By default, this box is **not** checked and it only verifies for well-formedness.

Resolve Externals

If this box is checked, all resolvable externals (namespaces, DTDs, entity references etc.) will be resolved at parse time. By default, this box is **not** checked and externals are not resolved.

Outputs

Payload

Node XML document with Envelope as document element. If there is an error, the document may be empty (but not NULL), and the 'XML Get Error Info' tool can be used to retrieve information about what failed).

Exit Paths

Success

This path is taken if the payload is successfully parsed. SOAP Action matches.

Empty Payload

This path is taken if the SOAP Payload is empty (XML document has no document element).

Wrong Action

This path is taken if the SOAP Action validation is enabled and action doesn't match.

Parse Error

This path is taken if a parse error occurred parsing the payload. Use 'XML Get Error Info'.

Failure

This path is taken for any other error. Use 'XML Get Error Info'.

SOAP Query Encoding Style

This SOAP tool matches a space separated list of URIs against the 'encodingStyle' attribute of the given element. If the element doesn't have an 'encodingStyle' attribute, the parent of the element is checked and so on, until an element with an 'encodingStyle' attribute is found. If that attribute contains any of the specified encoding style URIs, the tool returns through 'Success' and returns the style that was found.

Note: If the first 'encodingStyle' attribute found along the parent chain does not contain any of the specified styles, the search does not continue and the tool exits 'Not Found'.

Inputs

Element

Child Element of the SOAP envelope to query. If document node, the document element is queried.

Encoding Styles

Space separated list of URIs to match against the 'encodingStyle' attributes.

Outputs

First Style Found

Encoding style namespace that was found.

Element Of Style

XML node of the element in which the encoding style attribute was found.

Exit Paths

Found

This path is taken if the encoding style is successfully returned.

Not Found

This path is taken if the first 'encodingStyle' attribute found along the parent chain does not contain any of the specified styles.

Failure

This path is taken if the operation fails.

SOAP Send Response

This SOAP tool sends the specified payload as response to the sender of the request. To support transport specific features, the 'Transport Control Data' argument takes an XML node whose content will be sent back to the client. It can be used to send transport specific out-of-band data to the client. For example, for the HTTP transport it allows to set additional header fields or specify a special status code.

Inputs

SOAP Request

Handle of the SOAP request.

Payload

Node of the payload envelope to send back to the client. Must be document node or <Envelope> document element.

Transport Control Data

Optional. Node of an XML structure with additional transport specific control data. See remarks.

Exit Paths

Success

This path is taken if the response was sent successfully.

No Response

This path is taken if the request does not expect a response

Duplicate

This path is taken if the response for this request has already been sent

Failure

This path is taken if the operation fails. Check Payload node with XML Get Error Info

SOAP Set Element Type

In SOAP, the type of an argument or the return value is specified by the service description and doesn't need to be included in the payload. However, the service may define the type as **xsd:anyType**, for example for VARIANT types. In this case, the type must be included in the argument. For example, if a type of "double" is specified, an element will look as follows:

```
<Element xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xsd:double">1234.567</Element>.
```

Note: the type may be a user defined (complex) type. For example:

```
<ns1:Order xmlns:ns1="uri:my-order-type"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns1:Order">
<ns1:Product>Watchmacallit</ns1:Product>
<ns1:Quantity>7</ns1:Quantity>
<ns1:Price>19.99</ns1:Price>
</ns1:Order>.
```

Please refer to <http://www.w3.org/TR/xmlschema-0> or <http://www.w3.org/TR/xmlschema-2> for details on the XML Schema Datatypes.

Inputs

Element

Node of an element whose Schema instance type to set.

Type

String XSD type to declare for this element. See remarks for details. The argument may either be just the type name or have schema namespace prefix, such as **xsd:string**. If the type argument does not contain a prefix, **xsd** will be used.

Type Namespace

Namespace of the type. Optional.

Default: <http://www.w3.org/2001/XMLSchema>

XSI Namespace

XML Schema Instance namespace. Optional.

Default: <http://www.w3.org/2001/XMLSchema-instance>

XSI Namespace Prefix

Prefix of the schema instance namespace. Optional.

Default: **xsi**

Declare Namespaces in Envelope

Leave this box checked to declare the XSD and XSI namespaces in the Envelope element (actually, the document element is used, as this tool may be for other purposes than SOAP).

If any of the parent elements already has a NS declaration for a prefix and the namespace URI is different, the declaration will be added to the element, and not the Envelope.

Clear this checkbox to declare the XSD and XSI namespaces in the element itself.

Exit Paths

Success

This path is taken if element type is successfully set.

Failure

This path is taken if the operation fails.

SOAP Set Fault

This SOAP tool adds a <Fault> element to the envelope or replaces an existing one. If one of the mandatory fields (Fault Code, Fault Actor) is empty, the Failure path is taken and XML Get Error Info may be used on the Envelope node to query for error reasons.

Note: If the envelope already has a <Fault> element, the tool will remove the existing <Fault> element and replace it with the new element.

Inputs

Envelope

Envelope node of the SOAP payload. Can be a document node whose document element is <SOAP-ENV:Envelope> or the node is the element itself.

Fault Code

String to set as value of the <faultcode> element. String must not be empty.

Fault String

String to set as value of the <faultstring> element. Should be set to provide human readable information.

Fault Actor

String to set as value of the <faultactor> element. If argument is not specified, no <faultactor> element is added. Optional.

Create Detail Element

Leave this box checked to create an empty <detail> element. Clear this box if you don't want to create a <detail> element

Note: According to the SOAP spec, a <detail> element must be present if the fault is because the <Body> could not be processed successfully.

Preserve Body Elements

Leave this box unchecked to remove all existing body elements and replace with a <Fault> element. Check this box to leave the existing body elements and append <Fault> element as last child of <Body>.

Note: When sending a fault response to the client, only the <Fault> element is allowed in the body!

Outputs

Detail Element

Returns the node of the newly created <detail> element. If 'Create Detail Element' is False, a NULL node is returned.

Exit Paths

Success

This path is taken if the element is successfully added.

Failure

This path is taken if the operation fails.

Schedules

Get Schedules

This Schedules tool returns a list of Directory Services (DS) Keys representing schedules created in Interaction Administrator or Interaction Attendant. You narrow this list by specifying a profile and/or keyword.

Note: This tool is used by Interaction Attendant to return a list of schedules. While it is possible that this tool be used in other handlers, the handler author should have a solid understanding of CIC's IVR handlers, call flow, retrieving DS attributes, and branching based on the values of DS attributes. These tools are currently used "out-of-the-box" in the Interaction Attendant handlers. They may be used outside of Interaction Attendant in a future release.

See *Scheduling* in the *Interaction Administrator and Interaction Attendant* online documentation for more information on creating and using schedules.

Inputs

Schedule Type

Optional. Specify one of the following values:

- "Interaction Attendant"
Retrieves schedules created with Interaction Attendant.
- "Interaction Administrator"
Retrieves schedules created with Interaction Administrator
- "Both"
Retrieves schedules created in both Interaction Attendant and Interaction Administrator. This is the default value if you specify nothing in this parameter.

Profile

Optional. Retrieve schedules only for the specified Interaction Attendant profile. Specify the DS key path for the profile.

Keyword

Optional. Retrieve only Interaction Administrator schedules that contain the specified keyword in their title. Keywords are specified for each schedule in Interaction Administrator.

Schedule Class

Specify one of the following classes:

For this class:	Specify this value:
Normal	1
Operator	2
Outbound	3
Email	4

Outputs

Retrieved Schedules

A list of strings where each item in the list is a Directory Services (DS) path to a schedule that matches your input criteria. Portions of DS searched include \$CONFIG\Schedules and under \$SERVER\Attendant\<ProfileName> for each Profile Name.

Exit Paths

Success

This tool takes the Success exit path if the query is successful. The query may be successful if the search returns no schedules.

Failure

This tool takes the Failure exit path if it is unable to query Directory Services.

Related Topics

[Get Best Schedule](#)

Get Best Schedule

This Schedules tool returns the most appropriate schedule for a specified DateTime value.

Note: This tool is used by Interaction Attendant. While it is possible that this tool be used in other handlers, the handler author should have a solid understanding of CIC's IVR handlers, call flow, retrieving Directory Services (DS) attributes, and branching based on the values of DS attributes. These tools are currently used "out-of-the-box" in the Interaction Attendant handlers. They may be used outside of Interaction Attendant in a future release.

See *Scheduling* in the *Interaction Administrator and Interaction Attendant* online documentation for more information on creating and using schedules.

Inputs

Schedules To Compare

- A list of string value containing at least one entry. This list of string values is generated by the [Get Schedules](#) tool.

Date/Time

A DateTime value for which you want to find the best schedule. For example, suppose the Get Schedules tool returns a list of three schedules: SupportOpen, SupportClosed, and SupportHoliday. If the DateTime value that you specify here falls within the range specified for SupportOpen, then SupportOpen is chosen as the matching schedule. See Match Type for more information on this tool chooses schedules.

Match Type

Choose one of the options below:

- "First"
Retrieves the first matching schedule from the Schedules to Compare list and assigns it to the Best Schedule output parameter.
- "Best"
Retrieves schedules created with Interaction Administrator. The criteria for picking the best schedule are listed below:
 1. Is this schedule active? Yes, continue on. No, discard it.
 2. StartDate/EndDate comparison. These two values are configured in Interaction Attendant or Interaction Administrator. These two fields and the DatePeriod field define the effective date and time for comparison.
 3. DatePeriod comparison.
 4. StartTime/EndTime comparison.
 5. For any schedules that made it this far, CIC uses Periodicity to determine the best schedule. Periodicity is checked in the following order:
 - "Unplanned" - highest priority
 - "One Time"
 - "Repeat Yearly"
 - "Repeat Monthly"
 - "Repeat Weekly"
 - "Repeat Daily" - lowest priority
 6. If more than 1 schedule has made it to this point, the matching schedules have the same periodicity. The StartTime/EndTime of each schedule is used. The times that are closest to the start time would be returned.
 7. If a schedule is referenced in both places, the Interaction Attendant schedule subentry is returned.
 8. For an "Unplanned" schedule, a "live schedule" will be considered to be effective.

Outputs

Best Schedule

A string value containing the Directory Services path to the key containing the best schedule. If this value is empty, it means that no schedule matches.

Exit Paths

Success

This tool takes the Success exit path if the schedules are successfully queried. The query may be successful if the query returns no schedules.

Failure

This tool takes the Failure exit path if the Match Type value is empty or set to an invalid value.

Related Topics

[Get Schedules](#)

StatAlertServer

Alert Custom Handler Result

This StatAlertServer tool takes a result code from a custom tool and sends it back to StatAlertServer to indicate that the handler completed.

Inputs

Returns the result of an alert custom handler

The sequence ID provided to the initiator. This is used to correlate results with handler invocations in AlertServer.

Result Code (int)

The result code from the handler execution.

Exit Paths

Success

The tool takes the Success exit path if the result is successfully returned.

Failure

The tool takes the Failure exit path if the operation is not successful.

System

System Tools

The System tools allow you to create handlers that look up, create entries, update entries on the Directory Services. Some of these tools allow you to gather diagnostic information.

[Complete External Blind Transfer](#)

[Complete External Call](#)

[Complete External Call \(extended\)](#)

[Complete Intercom Blind Transfer](#)

[Complete Intercom Call](#)

[Delete DS Key](#)

[DialPlan Failure](#)

[Execute Shell Command](#)

[GetDsAttr](#)

[GetDsAttrs](#)

[GetDsKeys](#)

[Get Email Profile](#)

[Get Operator Profile](#)

[Get Profile](#)

[Is Distribution List Member](#)

[Keypad Map](#)

[Log Event](#)

[LookUp](#)

[Lookup List](#)

[PutDsAttr](#)

[PutDsAttrs](#)

[PutDSKey](#)

[Query Backup](#)

[Semaphore Lock](#)

[Semaphore Unlock](#)

[Send Custom Notification](#)

[Server Name](#)

[Sleep](#)

[Unique ID](#)

[Whitepages](#)

[WhitepagesAsynch](#)

[WhitePagesLocality](#)

Complete External Blind Transfer

This System tool transfers a call to a number outside your organization. This step has no exits because a handler or subroutine ends after this step has finished.

Note: This tool should only be used in the handlers System_InitiateManualDialing and System_InitiateCallRequest. This tool only works with call objects created by a telephone going off-hook, or by a CIC client user making a call (for example, a user clicking the Make Call button in Interaction Desktop). This tool fails if used in any other handlers or subroutines.

Inputs

Call Identifier

The unique identifier for a call to be transferred.

Telephone number

The phone number of the transfer recipient. A comma causes a two-second pause, and any numbers after the "/" symbol are dialed after the call is connected.

Name of the called party (optional)

This optional parameter can contain the name of the recipient. This parameter could also contain the results of a whitepages lookup.

Formatted Telephone Number

A formatted version of telephone number used in the search.

Lines Groups (empty list means any line)

The name of a particular line on which this call should be placed. An empty list indicates that this call may be placed on any available line.

List of dial strings to be used

This list of dial strings parallels the list of line groups from the parameter above. When line group from position one is attempted, dial string from position one is used, and so on. In the default shipping handlers, the DialPlanEX subroutine returns a list of dial strings that parallels the list of line groups. You can pass the list of dial strings from that subroutine into this parameter. A comma in a dial string causes a two-second pause, and any numbers after a "/" symbol are dialed after the call is connected.

Calling Party Number

This parameter passes a string of digits to be displayed as ANI or Caller ID on the phone system that receives the call. If you do not specify a value in this parameter, the Telephony Services subsystem uses the number entered in the Phone Number field in the Line Configuration container in Interaction Administrator.

Note: This only works for ISDN.

Calling Party Name

This parameter passes a string to display the name of the caller on the phone system that receives the call. If you do not specify a value in this parameter, the Telephony Services subsystem uses the name associated with the number entered in the Phone Number field in the Line Configuration container in Interaction Administrator.

Note: This only works for ISDN.

Use Putback (if available)

This Boolean, set appropriately by default by CIC according to the configured line's support for putting a call back on the originating system, is used to determine whether or not to use the putback feature if it is available for that line. If putback is available, then setting this parameter to True will cause the transfer to be attempted using the Putback operation. The transfer will be attempted using the originating SIP line if the originating SIP line is listed in the Dial Plan, regardless of the Dial Group preference. If the originating SIP line is not referenced in a matching Dial Group in the Dial Plan or the Putback operation fails within the TS Server, a conventional transfer will be attempted and will follow the Dial Group preference in the Dial Plan.

Note: We strongly recommend that you do not change the default value of this parameter unless you know exactly when and why you need to override the default system value. This value is normally passed through the Transfer Request Initiator, but it can be changed there as well.

Diversion Number

Used for forwarded SIP calls, this field specifies the destination/address to which the call was originally sent.

Diversion Name

A name for the SIP diversion address.

Diversion Reason

A number representing one of the SIP diversion reasons:

Value	Description
0	None
1	Unknown
2	Busy
3	No answer
4	Unavailable
5	Unconditional
6	Time of day
7	Do not disturb
8	Deflection
9	Follow-me
10	Out of service
11	Away
255	Other

Custom Headers

List of callAttr in custom headers. Use the format of key=value delimited by semicolon to pass to a SIP request. You cannot use a reserved header as a custom header. To use customized headers, enable the Allow Full Custom Headers server parameter. For more information, see [Optional General Server Parameters](#) in Interaction Administrator help.

Exit Paths

Success

The tool takes the Success exit path if the call is successfully transferred.

Busy

The tool takes the Busy exit path if the number called is busy.

Failure

The tool takes the Failure exit path if the operation is not successful.

Complete External Call and Complete External Call (extended)

These System tools call a number outside your organization.

Caution: Do not use these tools in any location other than where they are currently being used in the default handlers that ship with CIC. These tools only work with call objects created by a telephone going off-hook or by a CIC client user making a call (for example, a user clicking the Make Call button in Interaction Desktop). These tools fail if used in any other handlers or subroutines.

The following is a description of the parameters for the Complete External Call tool. See [Complete External Call \(extended\)](#) for more information about that tool.

Inputs

Call Identifier

The unique identifier for a call.

Telephone number

The phone number of the party to be called. A comma causes a two-second pause, and any numbers after the "/" symbol are dialed after the call is connected.

Name of Called Party

This optional parameter passes a string to display the name of the caller on the phone system that receives the call. If you do not specify a value in this parameter, the Telephony Services subsystem uses the name associated with the number entered in the Phone Number field in the Line Configuration container in Interaction Administrator.

Note: This only works for ISDN.

Formatted Telephone Number

This parameter passes a string of digits to be displayed as ANI or Caller ID on the phone system that receives the call. If you do not specify a value in this parameter, the Telephony Services subsystem uses the number entered in the Phone Number field in the Line Configuration container in Interaction Administrator.

Note: This only works for ISDN.

Lines Groups (empty list means any line)

This parameter takes a list of line groups (as configured in Interaction Administrator), or leave this parameter empty to use any available line group.

Exit Paths

Success

This step takes the Success exit path if there is an available line.

ISDN Cause Code Value

This tool takes the ISDN Cause Code Value exit path if this tool detected a busy signal. This tool then places a string value in the EIC_ISDNCauseValue call attribute. You would then use a selection step to branch based on the value of EIC_ISDNCauseValue. The

cause codes are numbers in string format, such as "42". You can view a list of cause codes and their explanations at <http://www.shout.net/~wildixon/telecom/isdn/cause-codes.html>.

No Lines

This tool takes the No Lines path if no outgoing lines are available.

Failure

This step takes the Failure exit path if there are no available lines. If this step fails, the call does not change [states](#).

Complete External Call (extended)

This System tool calls a number outside your organization.

Caution: Do not use this tool in any location other than where they are currently being used in the default handlers that ship with CIC. These tools only work with call objects created by a telephone going off-hook or by a CIC client user making a call (for example, a user clicking the Make Call button in Interaction Desktop). These tools fail if used in any other handlers or subroutines. These tools will not be supported in the next major release of CIC.

Inputs

Call Identifier

The unique identifier for a call.

Telephone number

The phone number of the party to be called. A comma causes a two-second pause, and any numbers after the "/" symbol are dialed after the call is connected.

Lines Groups (empty list means any line)

This parameter takes a list of line groups (as configured in Interaction Administrator), or leave this parameter empty to use any available line group.

List of dial strings to be used

This list of dial strings parallels the list of line groups from the parameter above. When line group from position one is attempted, dial string from position one is used, and so on. In the default shipping handlers, the DialPlanEX subroutine returns a list of dial strings that parallels the list of line groups. You can pass the list of dial strings from that subroutine into this parameter. A comma in a dial string causes a two-second pause, and any numbers after a "/" symbol are dialed after the call is connected.

Note: this parameter appears only in the Complete External Call (extended) tool.

Calling Party Number

This parameter passes a string of digits to be displayed as ANI or Caller ID on the phone system that receives the call. If you do not specify a value in this parameter, the Telephony Services subsystem uses the number entered in the Phone Number field in the Line Configuration container in Interaction Administrator.

Note: This only works for ISDN.

Calling Party Name

This parameter passes a string to display the name of the caller on the phone system that receives the call. If you do not specify a value in this parameter, the Telephony Services subsystem uses the name associated with the number entered in the Phone Number field in the Line Configuration container in Interaction Administrator.

Note: This only works for ISDN.

Custom Headers

List of callAttr in custom headers. Use the format of key=value delimited by semicolon to pass to a SIP request. You cannot use a reserved header as a custom header. To use customized headers, enable the Allow Full Custom Headers server parameter. For more information, see [Optional General Server Parameters](#) in Interaction Administrator help.

Exit Paths

Success

This step takes the Success exit path if there is an available line.

Busy

The tool takes the Busy exit path if the number called is busy.

No Lines

This tool takes the No Lines path if no outgoing lines are available.

Disconnect

The tool takes the Disconnect exit path if the call disconnects.

Failure

This step takes the Failure exit path if there are no available lines. If this step fails, the call does not change [states](#).

Complete Intercom Blind Transfer

This System tool transfers a call to another queue on the CIC system. Instead of a telephone number, specify the name of a User, Workgroup, or Station queue. This step has no exits because a handler or subroutine ends after this step has finished.

Note: This tool is for use only in system handlers that are provided by PureConnect.

Inputs

Call Identifier

The unique identifier for the call to be transferred.

Queue Identifier

The queue of the transfer recipient.

Name of the called party (optional)

This optional parameter can contain the name of the recipient.

This parameter could also contain the results of a Directory Services lookup.

Complete Intercom Call

This System tool places a call to another CIC queue and generates a Call To Non-System Queue event, which starts the [System_CallOfferingNonSystemQueue](#) handler. For this step you must specify the name of a User, Station, or Workgroup queue instead of a telephone number. This step has no exits because a handler or subroutine ends after this step has finished.

Note: This tool should only be used in the handlers `System_InitiateManualDialing` and `System_InitiateCallRequest`. This tool only works with call objects created by a telephone going off-hook, or by a CIC client user making a call (for example, a user clicking the Make Call button in Interaction Desktop). This tool fails if used in any other handlers or subroutines.

Inputs

Call Identifier

The unique identifier for a call.

Queue Identifier

The queue of the person being called.

Name of the called party (optional)

This optional parameter can contain the name of the recipient. This parameter could also contain the results of a Directory Services lookup.

DataManager Query

This System tool returns contacts from Data Manager. The tool can return the available contact sources and attributes or contacts from a particular source. Querying for sources would be used when querying a special source used as a catalog service. When querying for contacts from a particular source, the schema will allow for sparse data so that fields not requested do not cause the documents to be unnecessarily large.

The XML input parameter must be in a particular format to return the correct data.

Inputs

XML Node

Select an XML node for your query in one of the following formats:

Input Source Query Format

```
<query>
  <querymode>source</querymode>
</query>
```

Input Contact Query Format

```
<query>
  <querymode>contact</querymode>
  <rowlimit>1000</rowlimit>
  <owner>id</owner>
  <filter>ID=5</filter>
  <sortorder>ID</sortorder>
  <timeout>20000</timeout>
  <source>outlook</source>
```

```
<attributes>
  <field>FirstName</field>
  <field>LastName</field>
  ... additional fields
</attributes>
</query>
```

Outputs

XML Node

Select the an XML Node to output the results of the query. The results will be in one of the following formats:

Output Source Result Format

```
<sources>
  <source>
    <name>outlook</name>
    <attributes>
      <field>FirstName</field>
      <field>LastName</field>
      ... additional fields
    </attributes>
  </source>
  ... additional sources
</sources>
```

Output Contact Result Format

```
<contacts>
  <contact>
    <FirstName>First1</FirstName>
    <LastName>Last1</LastName>
    ... additional fields
  </contact>
  ... additional contacts
</contacts>
```

Exit Paths

Success

This path is taken if the query is successfully run.

Failure

This path is taken if the query fails.

Delete DS Key

This System tool deletes a key in Directory Services. This tool was designed to be used with the Interaction Attendant application, but could be used with similar applications. You can optionally choose to retain or delete the subkeys below the key you are deleting.

Caution: This tool could delete important information from your Windows NT Registry if used incorrectly, which could lead to serious problems with CIC. If you are not sure about using this tool, contact technical support.

Inputs

Directory Services Path

The key to be deleted, including the path. By default, the path value stored in the Directory Services Path server parameter is used. When using the default, you should only specify a key name and the default path to that key will be used.

Delete entries under this key? option

Select this option to delete all child keys under the parent key being deleted. Clear this option to retain the child keys.

Exit Paths

Success

This tool takes the Success exit path if the delete operation is successful.

Failure

This tool takes the Failure exit path if the operation is not successful.

Dialplan Failure

This System tool is only used in the Dialplan handler. It causes the dial (or transfer) request to fail with the indicated text displayed on the requesting client.

This step has no exits because a handler or subroutine ends after this step has finished.

Inputs

Call Identifier

The unique identifier for a call that has failed.

Status Text

The text message that is displayed in the CIC Client.

DID/DNIS Routing

This System tool takes the DNIS string of a telephone call and returns a scoped queue name based on the DID/DNIS configuration in Interaction Administrator.

Inputs

Telephone Number

The DNIS string to be routed.

Outputs

Queue Name

The scoped queue identifier.

Note: Station groups do not have associated queue identifiers, so if a match is found for a Station Group, the output is a scoped name: "Station Group:<stationgroup name>"

Exit Paths

Success

This path is taken if the scoped queue name is successfully retrieved.

Failure

This path is taken if the queue name is not retrieved.

DLL Function Call (Out of process)

This System tool calls a sample external DLL. Use this tool *as an example* when creating custom functionality (outside of CIC) without having to create a new Interaction Designer tool. You must include the ipdllfunction.h header file in any DLLs you create for use with CIC.

This tool works out of process. That is, the function call is moved to another process from the one that employed this tool. This way, if anything undesirable occurs as a result of the function call, the impact on the system will be minimal.

Inputs

DLL Name

The name of the DLL you want to call.

Function Name

The name of the function within the DLL you want to call.

Maximum time to wait for custom DLL function to finish

The number of seconds this tool will wait for the custom DLL function to finish before taking the Timeout exit path. Assigning a value of zero or any negative number to this parameter will cause the tool to wait indefinitely for the custom DLL function to finish.

Input 1

A string that contains an input value that the DLL uses.

Input 2

A string that contains an input value that the DLL uses.

Input 3

A string that contains an input value that the DLL uses.

Outputs

Output 1

A string that contains an output value from the DLL.

Output 2

A string that contains an output value from the DLL.

Output 3

A string that contains an output value from the DLL.

Exit Paths

Success

This step takes the Success exit path according to Boolean return code in the function. If the DLL can be loaded and the function located, users need to return 1 for this exit path to be taken or 0 for the Failure path.

Failure

This step takes the Failure exit path according to Boolean return code in the function.

DLL Not Found

This step takes the DLL Not Found exit path if the DLL could not be found or loaded. Basically the LoadLibrary call failed.

Function Not Found

This step takes the Function Not Found exit path if the specified function could not be located within the DLL.

Error

This step takes the Error exit path if the tool throws an uncaught exception.

Timeout

This step takes the Timeout exit path if the specified timeout period passes before the custom DLL function returns.

Execute Shell Command

This System tool executes any applications (.exe, .bat, .com) or files where an application is associated with an extension (.doc, .xls, .txt).

Caution: This tool does not differentiate between normal applications and applications that could be harmful to your system. For example, a user could use this tool to execute a `format.com c:` command to delete the contents of a hard drive. Be very careful with this tool.

Inputs

Directory Path

The path in which you want to run the application, or the path to the executable.

File Name

The name of the file to execute, including the extension.

Command Line Parameters

Any command line parameters the application should use when it executes.

Seconds to Wait

The timeout to use when running a command synchronously. See the **Run Synchronously** option.

Hide the command window

Check this box if you do not want a command window to appear when this step executes.

Run Synchronously

Check this box to run the command synchronously.

Note: Use this option sparingly because it will cause the invoking handler to wait until either the command completes or the timeout value is reached, which can consume a thread.

Outputs

Return Code

The number representing the results of the operation when run synchronously.

Exit Paths

Success

The file was found and the command to execute was given.

Failure

The file could not be found.

Wait Failure

If run synchronously and the specified timeout was reached.

External Handler Return

This System tool closes the handler called by the External Handler Call initiator and returns the result to the ExternalHandler C++ function that originally triggered the handler.

Inputs

Request Handle

A system generated identifier uniquely identifying the handler call that is generated by the [External Handler Call](#) initiator. This handle is used to correlate the External Handler Return invocation with the original External Handler Call.

Data

A list of strings containing information from the handler that will be passed back as an output parameter to the ExternalHandler C++ function.

Result

This is the result of the handler execution that will be returned to the ExternalHandler C++ function.

Exit Paths

Success

This path is taken if the handler data is successfully passed on to the C++ function.

Failure

This path is taken if the operation fails.

Feature Licensed?

This System tool takes the name of an uncounted license, such as the release of CIC or a Reverse White Page Lookup, and checks to see whether or not that licensed feature is available.

Inputs

Feature Name

The name of the feature being queried.

Exit Paths

Licensed

This path is taken if the specified feature is licensed.

Not Licensed

This path is taken if the specified feature does not have a license.

GetDSAttr

This System tool retrieves data from an attribute in a key in CIC's Directory Services (DS). You should use GetDSAttr when you can't retrieve the value with the Lookup tool. While the Lookup tool can only retrieve values from certain predefined keys, GetDSAttr can retrieve values from all keys, including keys you have created yourself. GetDSAttr cannot return any value in your registry, only those within HKEY_LOCAL_MACHINE\Software\Interactive Intelligence\CIC\Directory Services\Root.

There are three special DS attributes. They are SERVER, SITE, and CONFIG.

All DS tools only access the part of the registry that affects CIC.

"SERVER" accesses

"... Directory Services\Root\<<Site Name>\Production\<<Server Name>"

"SITE" accesses

"... Directory Services\ Root \<<Site Name>"

"CONFIG" accesses

"... Directory Services\ Root \<<Site Name>\Production"

The reason these special cases were created is so that handlers could be generically created to run on any CIC Server. If these weren't used, you would have to modify every DS tool anytime you wanted to use the handler on a different machine."

Example 1

Suppose you want to look up the value of the Default Voice Mail Recipient from within a handler. Since this value is not one of the key types that can be accessed with the Lookup tool, you must use the GetDSAttr tool. You should use two GetDSAttr steps; the first to retrieve the path to the Config key and the second to retrieve the value of an attribute within that key.

The first GetDSAttr step **Directory Services Path** should be set to "" (empty string) to tell GetDSAttr that you want to start in the Root directory with Directory Services. The **Directory Services Attribute** you want to retrieve is "CONFIG" since Default Voice Mail Recipient is a child key of CONFIG. These settings return the path to the Config directory on any CIC server, so this handler will be easily transportable to other CIC servers. In this example, we'll place that value in a variable called IsConfigPath.

The second GetDSAttr step **Directory Services Path** should be set to GetHead(IsConfigPath) & "\\Configuration". This appends the path info onto the specific key name you want to search within. The **Directory Services Attribute** you want to retrieve is Default Voice Mail Recipient. Remember that the attribute name is context-sensitive and must match the case displayed in the registry. In this example, the first entry in the returned list is the default voice mail recipient mailbox.

Another use for this tool is to [retrieve the value of a custom server or system parameter](#).

Inputs

Directory Services Path

The path to the Directory Services Key being accessed and the name of the key. DsPath is the default name for this variable.

Directory Services Attribute

The attribute to lookup in the specified Directory Services Key.

Outputs

List of Attribute Values

A variable of type ListOfString that contains a list of values retrieved from the specified attribute. Null values are not retrieved.

Exit Paths

Success

This step takes the Success exit path if the Attribute exists and the path is correct.

Failure

This step takes the Failure exit path if the Attribute does not exist or if the path to the Attribute is incorrect.

GetDSAttrs

This System tool retrieves a list of all the attributes in a key in CIC's Directory Services. Also returned are any values associated with those attributes. You should use GetDSAttrs when you can't retrieve the list of values with the LookupList tool. While the LookupList tool can only retrieve values from certain predefined keys, Det DS Attrs can retrieve values from all keys, including keys you have created yourself. GetDSAttrs cannot return any values in your registry, only those within HKEY_LOCAL_MACHINE\Software\Interactive Intelligence\CIC\Directory Services\Root.

There are three special DS attributes. They are SERVER, SITE, and CONFIG.

All DS tools only access the part of the registry that affects CIC.

"SERVER" accesses "...Directory Services\Root\\Production\

"SITE" accesses "...Directory Services\Root\

"CONFIG" accesses "...Directory Services\Root\\Production"

Inputs

Directory Services Path

The path to the Directory Services Key being accessed and the name of the key. DsPath is the default name for this variable.

Outputs

List of Directory Services Attributes

This is the list of all the attributes for the specified Directory Services Key. DsAttrList is the default name of the variable that contains this value.

List of Attribute Values

This is a list of corresponding values associated with the specified attributes returned in the previous parameter. If more than one value exists for a single attribute, only the first is returned.

Exit Paths

Success

This step takes the Success exit path if the key and Attributes exist and the path is correct.

Failure

This step takes the Failure exit path if the key or its Attributes do not exist or if the path to the Attribute is incorrect.

Get Ds Keys

This System tool step returns a list of all the child keys (and their paths) for a specific parent key in CIC's Directory Services.

Inputs

Directory Services Path

The path to the parent Directory Services Key containing the child keys.

Outputs

List of Directory Service Keys

This is the list of all the child keys for a specific parent key.

List of Directory Services Paths

This is a list of corresponding paths for the Directory Services Keys returned in the previous parameter.

Exit Paths

Success

This step takes the Success exit path if the key name is correct or the path to the key is correct.

Failure

This step takes the Failure exit path if the key name is incorrect or the path to the key is incorrect.

Get DS Parameter

This System tool queries a server or system parameter directly. If the parameter does not exist, this tool can also be used to create it.

Inputs

Parameter Source

This literal string must be either "SERVER" or "SYSTEM."

Parameter Name

The name of the parameter being queried.

Add Parameter?

Set this to True if you want the server parameter to be added if it does not exist already.

Default Value

The default value of the specified parameter. This value will be assigned to the parameter if it is being created by this tool.

Outputs

Value

The value of the parameter.

Exit Paths

Success

This path is taken if the value for the specified parameter is retrieved.

Failure

This path is taken if the operation fails.

Get Email Profile

This System tool scores each profile under the active configuration and returns the highest scored profile or the default profile if none of the profiles matches the input email information. This tool will look at all active profiles in Attendant and score each profile based on the email and the Incoming Email Selection data for each profile.

Scores for the other inputs are used to further refine any matches, as each profile could have 0, 1 or more pieces of matching information. The 1100, 1010, 1001 values are used to give higher weights to From Exact Match vs From Wildcard Match. Definitions for these types of matches can be found in Interaction Attendant.

If a profile has more than one piece of scoring information, that profile's score is determined by adding those scores together. For example, if one profile has both a From Exact Match and From Wildcard Match, the scores for both matches will be added together to determine that profile's score.

Inputs

Mailbox

The name of the mail box.

Email Cookie

Cookie of the incoming message.

From Exact Match Score

Scoring value to assign to a profile with an exact From Match.

From Wildcard Match Score

Scoring value to assign to a profile with an wildcard From Match.

Outputs

Profile

The profile with the highest score. If no profiles received any scores, the default profile will be returned.

Exit Paths

Success

This path is taken if the profile is successfully returned.

Failure

This path is taken if the operation fails.

Get Operator Profile

This System tool scores each operator profile under the active configuration based on the specified User Queue, Workgroup Queue, or Profile information and returns operator profile with the highest matching score.

Inputs

User Queue

The user queue specified to match the operator profile.

Workgroup Queue

The workgroup queue specified to match the operator profile.

Profile

The profile specified to match the operator queue.

Outputs

Operator Profile

The operator profile with the highest matching score.

Exit Paths

Success

This path is taken if the operator profile is successfully returned.

Failure

This path is taken if the operation fails.

Get Profile

This System tool scores each profile under the active configuration and returns the highest scored profile or the default profile if none of the profiles matched the line, DNIS, or ANI information. Line, DNIS, and ANI information comes from the Interaction Attendant Profile configuration, "Incoming Call Selection." This tool will look at all active profiles in Attendant and score each profile based on the Line, DNIS, and ANI of the call and the Incoming Call Selection data for each profile.

Note: Starting in CIC 4.0 SU3, this tool supports complex SIP/Tel URI addresses when used to match Interaction Attendant addresses.

Scores for the other inputs are used to further refine any matches, as each profile could have 0, 1 or more pieces of matching information. The 1100, 1010, 1001 values are used to give higher weights to Exact Match vs Range Match vs Wildcard Match. Definitions for these types of matches can be found in Interaction Attendant.

If a profile has more than one piece of scoring information, that profile's score is determined by adding those scores together. For example, if one profile has both an ANI Exact Match and a DNIS Wildcard Match, the scores for both matches will be added together to determine that profile's score.

This tool will match on SIP addresses. For example if had a DNIS match of 8723000 it would also match on a SIP call addressed to sip:8723000:5060:

You may also enter text based DNIS entries such as johndoe in a profile. This would then match for a SIP call addressed as sip:dukes@inin:5060:

Inputs

Line

The Line being scored.

DNIS

The DNIS being scored.

ANI

The ANI being scored.

DNIS Exact Match Score

Scoring value to assign to a profile with an exact DNIS match.

DNIS Range Match Score

Scoring value to assign to a profile with a DNIS Range Match.

DNIS Wildcard Match Score

Scoring value to assign to a profile with a DNIS Wildcard Match.

ANI Exact Match Score

Scoring value to assign to a profile with an exact ANI match.

ANI Range Match Score

Scoring value to assign to a profile with an ANI Range Match.

ANI Wildcard Match Score

Scoring value to assign to a profile with an ANI Wildcard match.

Trunk Exact Match Score

Scoring value to assign to a profile with an exact Trunk match.

Trunk Group Match Score

Scoring value to assign to a profile with a Trunk Group match.

Outputs

Profile

The profile with the highest score. If no profiles received any scores, the default profile will be returned.

Exit Paths

Success

This path is taken if the profile is successfully returned.

Failure

This path is taken if the operation fails.

Get Structured Parameter

This System tool is used to get a structured parameter. Structured parameters are organized by groups and have a type associated with them. The type can be used to have encrypted data in DS. This allows you to store passwords for other systems in DS without them being in readable form in the registry.

With this tool, you specify a parameter name and a group name and get the value back. The value will be a single string if the type is string, and a list of strings if the type is multi-string.

Inputs

Parameter Group

The group name of the structured parameter.

Parameter Name

The name of the structured parameter.

Outputs

Parameter Value

The value of the structured parameter.

Exit Paths

Success

This path is taken if the structured parameter value is successfully returned.

Cannot Find Parameter

This path is taken if the specified parameter cannot be found on the server.

Error

This path is taken if some other error occurs.

Keypad Map

This System tool takes a string and maps each character to its corresponding character from a telephone keypad. The numeric equivalent is stored as a string value. For example, the character 'b' would be mapped to the keypad equivalent of '2'. 'Q' is mapped to '7', and 'Z' is mapped to '9'.

Inputs

String to be mapped to keypad equivalent

This is the string to be converted.

Map punctuation to "*"?

Set this parameter to true to map all punctuation to the character "*". Set this parameter to false if you want punctuation to be ignored.

Outputs

Keypad equivalent string

This string value contains the numeric equivalent to the keypad presses.

Exit Paths

Next

This step always takes the Next exit path.

Load Localized String

This System tool retrieves a string from a specified DLL in the specified language, allowing customers to retrieve localized strings from custom DLLs.

Inputs

Module name (full path)

The full path to the DLL that contains the string you want to load.

Unload the module when exit

If False, keeps the DLL in memory after use. If True, unloads it. If the tool is executed frequently, it is more efficient to leave it loaded.

Language

The language/sub-language identifiers to retrieve (for example, "en-us").

String Identifier

The integer identifier of the string to retrieve.

Outputs

Localized String

The retrieved string value.

Exit Paths

Success

This path is taken if the string is successfully retrieved.

Failure

This path is taken if the DLL could not be loaded, the string identifier did not exist, or a value for the specified language does not exist.

Log Event

This System tool writes a message to the Windows NT event log.

Inputs

Log Type

The type of message to be written to the log. Legal values are "Error," "Warning," and "Informational."

Message to be placed in the log

The text of the message to be placed in the log.

Exit Paths

Next

This step always takes the Next exit path.

LookUp

This System tool retrieves information from Directory Services, the storehouse for information about users, workgroups, stations, lines, line groups, and DNIS/DID routing, server parameters, and wrap-up codes.

This information stored in Directory Services comes from several sources. Some of it is entered through Interaction Administrator when someone configures a new user, workgroup, station, line or line group. Some of the information comes from Exchange (or Domino) server, such as email addresses and email aliases. Some of it comes from changes made on a CIC Client configuration page.

The Lookup tool provides access to Directory Services from within a handler. For example, in the Voice Mail handler, a Lookup step retrieves the email address associated with the queue on which the call resides. In another example, a Lookup step might retrieve the extension associated with a Workgroup queue.

Note: Some User Rights attributes (e.g., View Workgroup Queue List, View Station Queue List, Modify Attendant Configurations, Modify Line Queue List, etc.) take longer to be updated and propagated through the system than all other attributes. Some administrative changes to these user rights might not be immediately visible to the Lookup tools. If timing is important with the Lookup tool steps, use the IC Change Notification Monitor initiator to monitor for User Rights changes before using the Lookup tools.

To use the Lookup tool, you need to know which *key* in Directory Services contains the information you want, and you need to know the *value of one of the attributes within that key*. With that key and attribute information, you can look up any attribute in Directory Services.

Example 1 (searching with a scoped queue identifier)

Suppose you want the email address for the user queue called "StephenS". You know the key is "User", and the value of "Queue Identifier" is "User Queue:StephenS". Since you have these two items, you can retrieve the email attribute associated with "UserQueue:StephenS".

For this example, the Inputs page would have the following values:

Key Type:	"User"
Search Attribute Type:	"Queue Identifier"
Value to use in search:	"User Queue:StephenS"
Perform leading substring comparison?:	true
Perform case insensitive comparison?:	true
Compare based on Keypad Mappings?:	false
Attribute type:	"Mailbox"

To make this step more flexible, you could substitute "StephenS" with a string variable whose value could be determined when the handler runs.

The outputs page would have the following values:

Attribute Value:	MailboxAttrib
Number of matching entries found:	MatchCount

MailboxAttrib is a variable that will contain returned email address. With these settings, this step will try to match any "User" key "Queue Identifier" attribute whose value is "User Queue:StephenS". If the tool finds a match it will return the "Mailbox" attribute value into a variable called MailboxAttrib.

Example 2 (searching without a scoped queue identifier)

You could also look up the Extension for the workgroup "TechSupport". You know the key is Workgroup and the Queue Identifier attribute is "TechSupport". You can retrieve the "Extension" attribute associated with the "Queue Identifier" attribute. In this example, Foo is a string variable whose value is "Tech Support". This step then returns the extension associated with the Queue Identifier contained in Foo.

For this example, the Inputs page would have the following values:

Key Type:	"Workgroup"
Search Attribute Type:	"Queue Identifier"
Value to use in search:	Foo
Perform leading substring comparison?:	true
Perform case insensitive comparison?:	true
Compare based on Keypad Mappings?:	false
Attribute type:	"Extension"

In this example, Foo is a string variable whose value is "TechSupport". This step then returns the extension associated with the Queue Identifier contained in Foo.

The outputs page would have the following values:

Attribute Value:	WorkgroupQueueExtension
Number of matching entries found:	MatchCount

WorkgroupQueueExtension is a variable that will contain the value of the returned extension.

Inputs Page

Key Type

The name of the key in CIC Directory Services you want to search. When you type a key, *its name must be enclosed in quotes*.

For example, if you were looking up an email address for a specific user, you would type "User" in this field. The keys that can be searched are:

- "DnisDid"
- "Line"
- "LineGroup"
- "StationGroup"
- "User"
- "Workgroup"
- "Workstation"

Search Attribute Type

The known attribute type you will use as the basis for your search. If you knew the Queue Identifier of a User and you wanted to return some attribute for that Queue, you would type "Queue Identifier" in this parameter.

Refer to the [Attributes that can be looked up in keys](#) for a list of attributes that can be looked up.

Value to use in search

The known attribute value that you are using as the basis for your search. For example, if you wanted to return some attribute for the "Queue Identifier" John Doe, you would type "User Queue:John Doe" in this field. In most cases, you'll use a variable in this field to make the Lookup step flexible. The value should be a regular expression value.

Note: When the Value to Use in Search is a queue name, you should always use a scoped queue name, such as User Queue:Stephens, Workgroup Queue:Marketing, Line Queue:ISDN1, or Station Queue:Fax1.

Perform leading substring comparison

Set this parameter to false if you want to search for an exact match between the search value and the attribute value in Directory Services. If this parameter is set to true, the comparison will succeed if the leading characters of the attribute string are equal to the comparison string. For example, if you are searching for the number 123, attributes with a value of 123, 1234 or 12345 are matches. 51234 or 4321 are not matches.

Perform case insensitive comparison?

Set this parameter to true if you are searching for a match including case. Set this value to false if you are not concerned with the upper or lower case.

Compare based on keypad mappings?

Keypad mappings are the letters associated with the numbers on a telephone keypad. These letters are printed on the keys of most telephones. For example, the character 'b' would be mapped to the keypad equivalent of '2'. 'Q' is mapped to '7', and 'Z' is mapped to '9'. By default this parameter is false so that keypad mapping is not performed.

Use Regular Expression for Lookup

Set this parameter to true if the value to use in the search is a regular expression value. Note that certain characters in regular expression values require escape sequences.

Attribute type

The attribute type you want the Lookup to return. For example, if you want to return a user's email address, you would type "Mailbox". Refer to the [Attributes that can be looked up in keys](#) for a list of attributes that can be returned.

Outputs Page

Attribute Value

The variable that will contain the value returned from the search. This will be null if zero, or more than one, matches are found.

Number of matching entries found

The number of matches found.

Exit Paths

Success

This step takes the Success exit path if the Lookup step finds only one match.

Failure

This step takes the Failure exit path if the Lookup step finds no match or if the input parameters contain invalid values. It will also take the Failure path if it cannot access Directory Services.

Ambiguous

This step takes the Ambiguous exit path if the Lookup step finds more than one match. If the Perform Leading Substring Comparison parameter is set to true and more than one match is found, this step takes the Ambiguous path, but the Attribute Value contains the value of the exact match.

LookupList

This System tool returns all of the non-null values for a specific attribute in a specific key. These values are placed in a variable of type ListOfString. Refer to the [Attributes that can be looked up in keys](#) topic for a list of attributes that can be looked up.

If you are just looking up a single attribute value, such as a user's email address, or a workgroup's extension, use the [Lookup](#) tool. Lookup and LookupList have similar functionality, so refer to the [Lookup](#) tool online help for examples of looking up values in Directory Services Keys.

Inputs

Key Type

The name of the key in CIC Directory Services you want to search. When you type a key, *its name must be enclosed in quotes*. For example, if you were looking up an email address for a specific user, you would type "User" in this field. The keys that can be searched are:

- "DnisDid"
- "Line"
- "LineGroup"
- "Server Parameter"
- "StationGroup"
- "User"
- "Workgroup"
- "Workstation"
- "Wrap-up Code"

Note: Some User Rights attributes (e.g., View Workgroup Queue List, View Station Queue List, Modify Attendant Configurations, Modify Line Queue List, etc.) take longer to be updated and propagated through the system than all other attributes. Some administrative changes to these user rights might not be immediately visible to the Lookup tools. If timing is important with the Lookup tool steps, use the IC Change Notification Monitor initiator to monitor for User Rights changes before using the Lookup tools.

Attribute Type

The specific attribute in the key type you want to search. Refer to the [Attributes that can be looked up in keys](#) topic for a list of attributes that can be looked up. *If you type an attribute name as a literal value, it must be enclosed within quotes.*

Sort Lists by Value?

False will sort the lists lexicographically by key value. True will sort lexicographically by attribute value.

Outputs

Key Type

A variable of type ListofString that contains a list of child key names retrieved from the search. Null values are not retrieved.

Attribute Value

A variable of type ListofString that contains a list of values retrieved from the specified attribute. Null values are not retrieved.

Exit Paths

Success

This step takes the Success exit path if the path to the key is valid and the attribute type and key type are valid.

Failure

This step takes the Failure exit path if the path to the key is invalid, or if the attribute or key type is invalid.

Lookup List Extended

This System tool is similar to the [Lookup List](#) that uses and returns lists of strings rather than single string value. This tool returns parallel lists of users and requested attribute values, based on substring, case insensitive and/or keypad matches.

The parallel lists output by this tool are sorted based on the "Sort lists by value?" input and filtered based on the criteria set in the three filtering inputs, "Filter Columns", "Filter Values", and "Filter Types". Note that these three filter parameters must all have the same number of elements in their lists. If their length is zero, then the resulting parallel lists will not be filtered.

Please refer to the [Lookup](#) tool online help for examples of looking up values in Directory Services Keys.

Inputs

Key Type

The name of the key in CIC Directory Services you want to search. When you type a key, its name must be enclosed in quotes. For example, if you were looking up an email address for a specific user, you would type "User" in this field. The keys that can be searched are:

- "DnisDid"
- "Line"
- "LineGroup"
- "Server Parameter"
- "StationGroup"
- "User"
- "Workgroup"
- "Workstation"
- "Wrap-up Code"

Note: Some User Rights attributes (e.g., View Workgroup Queue List, View Station Queue List, Modify Attendant Configurations, Modify Line Queue List, etc.) take longer to be updated and propagated through the system than all other attributes. Some administrative changes to these user rights might not be immediately visible to the Lookup tools. If timing is important with the Lookup tool steps, use the IC Change Notification Monitor initiator to monitor for User Rights changes before using the Lookup tools.

Attribute Type

The specific attribute in the key type you want to search. Refer to the [Attributes that can be looked up in keys](#) topic for a list of attributes that can be looked up. *If you type an attribute name as a literal value, it must be enclosed within quotes.*

Sort Lists by value?

False will sort the lists lexicographically by key value. True will sort lexicographically by attribute value.

Value to use in search

The known attribute value that you are using as the basis for your search. For example, if you wanted to return some attribute for the "Queue Identifier" John Doe, you would type "User Queue:John Doe" in this field. In most cases, you'll use a variable in this field to make the Lookup step flexible.

Note: When the Value to Use in Search is a queue name, you should always use a scoped queue name, such as User Queue:Stephens, Workgroup Queue:Marketing, Line Queue:ISDN1, or Station Queue:Fax1.

Perform leading substring comparison?

Set this parameter to false if you want to search for an exact match between the search value and the attribute value in Directory Services. If this parameter is set to true, the comparison will succeed if the leading characters of the attribute string are equal to the comparison string. For example, if you are searching for the number 123, attributes with a value of 123, 1234 or 12345 are matches. 51234 or 4321 are not matches.

Perform case insensitive comparison

Set this parameter to true if you searching for a match including case. Set this value to false if you are not concerned with the upper or lower case.

Compare based on Keypad mappings?

Keypad mappings are the letters associated with the numbers on a telephone keypad. These letters are printed on the keys of most telephones. For example, the character 'b' would be mapped to the keypad equivalent of '2'. 'Q' is mapped to '7', and 'Z' is mapped to '9'. By default this parameter is false so that keypad mapping is not performed.

Use Regular Expression for Lookup

Set this parameter to true if the value to use in the search is a regular expression value. Note that certain characters in regular expression values require escape sequences.

Filter Columns

This is a list of strings of additional Directory Services attributes (e.g., user attributes in the User key) used to refine your lookup.

Filter Values

This list of strings provides attribute values to search for in the corresponding "Filter Columns" list. If the Filter Columns parameter contains two attributes to search for, this Filter Values field should contain two values, each one corresponding to a Filter Column (attribute) and listed in the same order. If you do not specify the same number of values as columns (attributes), an empty string will be substituted for the missing value(s) and the extra columns will be ignored. The system will log an error message in the trace log if the number of values does not match the number of columns (attributes).

Filter Types

This list of booleans of True or False corresponds to each of the Filter Columns and Filter Values elements and controls whether the lookup match is inclusive or exclusive for each element.

The lookup for each named column (attribute) and value can be inclusive (True) or exclusive (False). That means that an inclusive

filter will return only those rows where a Filter Value is found in the specified Filter Column (attribute). An exclusive filter will return only those rows where the Filter Values are NOT found in the specified Filter Column. that filter column values are not equal to the filter value of that column.

Outputs

Key Type

A variable of type ListOfString that contains a list of child key names retrieved from the search. Null values are not retrieved.

Attribute Value

A variable of type ListOfString that contains a list of values retrieved from the specified attribute. Null values are not retrieved.

Exit Paths

Success

This step takes the Success exit path if the path to the key is valid and the attribute type and key type are valid.

Failure

This step takes the Failure exit path if the path to the key is invalid, or if the attribute or key type is invalid.

Example

Suppose you want to find out which users have a 'Title' of 'Support Engineer', live in Texas and whose status is not 'Do Not Disturb'.

Set the following inputs to the Lookup List Extended toolstep:

- Key: User
- Attribute type: 'title'
- Sort Lists by value: (This value is unimportant for this example)
- Value to use in search: 'support engineer'
- Perform Leading substring comparison: No
- Perform case insensitive comparison: yes
- Compare based on keypad mappings: no
- [Note that these inputs expect lists of strings or booleans, but this shows commas for display purposes.]
- Filter columns: 'StateOrProvince', 'Status Text' /// These are the DS user attributes to search
- Filter Values: 'TX', 'Do Not Disturb' /// These are the attribute values to match
- Filter Types: True, False /// True means match "TX" and False means match any status except "Do Not Disturb"

Make External Handler Call

This System tool makes an external handler call by firing the External Handler Call initiator on another Notifier server. This allows for a Notifier server to initiate a handler on another Notifier server running its own IP processor and handlers, and to get results from that handler. This allows for the two IP processors to execute tasks and handlers in a synchronous manner.

When executing on the local Notifier server, this tool has the same effect as calling the C++ ExternalHandler function.

Inputs

Server

The ID of the Notifier server on which the external handler is published.

Login

The login ID required for accessing the Notifier server.

Password

The password associated with the login being used.

Operation

The name of the handler operation that will be fired. This is passed to the External Handler Call initiator on the remote IP processor.

Data

A list of strings containing information from the handler that is passed as an input parameter to the External Handler Call initiator on the remote IP processor.

Timeout

The amount of time, in milliseconds, that this tool will wait for a response from the external server. If there is no response from the server in the allotted time, this tool will take the "Failure" exit path.

Note: This timeout does not include the time required to connect to the remote server. It applies only to the time it takes for the remote server to execute a handler and return a result once the connection is established.

Outputs

Data

The list of strings containing output information from the handler executed on the remote IP processor.

Result

The result of execution of the handler on the remote IP processor.

Diagnostic

This output is used in determining the cause for failure when the result of the external handler is False. This string contains information to help diagnose the failure that occurred on the remote IP processor when executing the external handler.

Exit Paths

Success

This path is taken if the external handler is successfully called.

Failure

This path is taken if the operation fails. This could occur if the remote server name is incorrect or if the login and/or password used are invalid.

OCR Installed

Important: The system no longer supports OCR tools.

This System tool determines whether the OCR Server is installed on your CIC Server. Typically this tool is used before proceeding with OCR processing, which would fail if the OCR Server is not installed.

Exit Paths

Success

This tool takes the Success path if the OCR Server is installed.

Failure

This tool takes the failure path if the OCR Server is not installed.

Parse String RegEx

This System tool allows the parsing of a string based on a standard regular expression. This could be used to check and match IP, email, or SIP addresses within a string.

Inputs

String

The string to parse.

Expression

A string containing the regular expression with which the string is to be parsed. The regular expression uses the following escape sequences:

"\+"	Represents the one or more repetition operator
"\?"	Represents the zero or one repetition operator
"\[\" and "\]"	Used to specify character ranges (sets)
"\{" and "\}"	Used for bounded repetitions
"\" and "\)"	Used to group sub-expressions
" "	Represents the alternation operator

Note: To get a single slash in the actual string at runtime, you need to enter double slashes. For example, to get the string "+" you need to enter "\\+" in the Interaction Designer editor.

The expression syntax supported in this tool is based on the Boost Library's Perl regular expression syntax, which is available online at: <http://www.boost.org> (Search the **Documentation** link for: "Perl regular expression syntax")

Case Sensitive

When this checkbox is selected, case-insensitive comparisons are used. Clear this checkbox if you do not want the comparisons to be case-sensitive.

Outputs

Result

Text string containing the results of the expression applied to the parsed string.

Exit Paths

Success

This path is taken if the string is successfully parsed.

Failure

This path is taken if the regular expression did not match the input string.

Put Ds Attr

This System tool step adds (or replaces) a list of values to a single attribute in a specific CIC Directory Services key.

Caution: Any changes you make to values within Directory Services can cause CIC to fail. Many of the settings CIC uses to run correctly are stored in Directory Services. When viewing your CIC Server registry, always make a backup copy before you begin exploring. For the following instructions, you should only need to view the location of certain keys within Directory Services. If you have any questions about viewing your Window's NT registry, contact your system administrator or technical support representative.

Inputs

Directory Services Path

This is the path to, and the name of, the Directory Services key containing the attribute you want to modify.

Directory Services Attribute

The name of the attribute to be modified in the specified Directory Services key.

List of Attribute Values

This list of strings contains the values that are added to the attribute. An attribute that is given a value of "" (i.e., no value) as its only value will be removed from the entry.

Append Attribute Values

Set this parameter to true to append the new value to the existing values. Set this parameter to false to replace the old values with the new values.

Exit Paths

Success

This step takes the Success exit path if the path to the key is valid and the attribute type and key type are valid.

Failure

This step takes the Failure exit path if the path to the key is invalid, or if the attribute or key type is invalid.

Put Ds Attrs

This System tool step adds a list of values to a list of attributes in a specific CIC Directory Services key.

Caution: Any changes you make to values within Directory Services can cause CIC to fail. Many of the settings CIC uses to run correctly are stored in Directory Services. When viewing your CIC Server registry, always make a backup copy before you begin exploring. For the following instructions, you should only need to view the location of certain keys within Directory Services. If you have any questions about viewing your Window's NT registry, contact your system administrator or technical support representative.

Inputs

Directory Services Path

This is the path to, and the name of, the Directory Services key containing the attributes you want to modify.

List of Directory Services Attributes

The list containing the attributes to be modified in the specified Directory Services key.

List of Attribute Values

This is the list of values for the attributes.

Append Attribute Values

Set this parameter to true to append the new values to the existing values. Set this parameter to false to replace the old values with the new values.

Exit Paths

Success

This step takes the Success exit path if the path to the key is valid and the attribute types and key type are valid.

Failure

This step takes the Failure exit path if the path to the key is invalid, or if the attribute types or key type is invalid.

Put Ds Key

This System tool creates a Directory Services key.

Inputs

Directory Services Path

This is the path to, and the name of, the Directory Services key to be created.

Class of entry to be created

The type of Directory Services key to be created. The valid types are:

- Accumulator
- Accumulators
- Action

- Actions
- Audio Sample
- Audio Samples
- Caller ID
- Caller ID Map
- Caller ID Maps
- Configuration
- Configuration Set
- Data Manager
- Data Source
- Data Sources
- Default User
- Default Users
- Designer
- Fax
- Fax Driver
- Fax Drivers
- Fax Group
- Fax Groups
- Global Variable
- Global Variables
- Handler
- Handlers
- Initialization Function
- Initialization Functions
- Initialization Handler
- Initialization Handlers
- IP
- Line
- Line Group
- Line Groups
- Lines
- Pager
- Parameter
- Parameters
- Remote Administrator
- Remote Administrators
- Report
- Report Log
- Report Logs
- Reports
- Server
- Server Accumulator
- Server Accumulators
- Server IP
- Server Report Log
- Server Report Logs
- Server Table
- Server Tables
- Site
- Skill
- Skills
- Status Message
- Status Messages
- Subroutine
- Subroutines
- User

- Users
- Web Form
- Web Forms
- Web Initiator
- Web Initiators
- Workgroup
- Workgroups
- Workstation
- Workstations

Exit Paths

Success

This step takes the Success exit path if the path to the key is valid.

Failure

This step takes the Failure exit path if the path to the key is invalid.

Put External Password

This System tool is used to give login information to an external service such as a mainframe.

Inputs

Password Id

The unique identifier for this set of login information.

Server Name

The server receiving the login information.

Account Name

The account for which the login information is needed.

Password

String containing the password information.

Warning: If a literal value is used, the password is plainly visible. For this reason, we recommend creating a handler that uses a custom notification to pass the password in as a parameter from the command line. This way, anyone looking at the handler would see only that the passwords are passed in, and not the actual passwords.

Custom String Value

This parameter contains whatever additional information is needed to log in.

Exit Paths

Success

This path is taken if the login information is successfully passed to the external service.

Failure

This path is taken if the operation fails.

Put Structured Parameter

This System tool is used to put a structured parameter onto the server. Structured parameters are organized by groups and have a type associated with them. The type can be used to have encrypted data in Directory Services. This allows you to store passwords for other systems in Directory Services without them being in readable form in the registry.

With this tool, you specify a group name, and parameter name, a value, and whether or not you want it obfuscated.

Inputs

Parameter Group

The group name of the structured parameter.

Parameter Name

The name of the structured parameter.

Parameter Value

The value of the structured parameter.

Obfuscate the value

Check this box if you want to obfuscate the value of the structured parameter.

Exit Paths

Success

This path is taken if the structured parameter is successfully placed on the server.

Error

This path is taken if the operation fails.

Query Backup

This System tool determines whether or not a CIC server is part of a switchover environment and whether or not it is a backup server. It does this by examining the machine on which it is running. Use this tool to have a handler decide whether it should perform functionality only on the primary server or backup server. For example, you might not want to have the handler send an email if running on the backup server. This tool allows you to build a patch in a handler that is dependent on the server being either primary or backup.

Inputs

Maximum time to wait for response (In seconds)

Enter the number of seconds to wait before the backup server query fails.

Outputs

Backup

This Boolean value is:

TRUE if the CIC Server is a backup server.

FALSE If the CIC Server is a primary server.

Exit Paths

Success

This exit path indicates that the CIC server is in a switchover environment. The Backup output flag can either be TRUE or FALSE.

Failure

This exit path indicates that the server is not part of a switchover environment.

Query Conversation ID First Agent

This System tool returns the first agent and workgroup who handled the email conversation.

Note: This tool uses Interaction Tracker data, so you must have the Interaction Tracker Access or Recorder license for the tool to work. If the license is not available, the tool returns empty values.

Inputs

Email Conversation ID

The identifier for the email conversation for which you want to find the first user and workgroup.

Outputs

First ICUser

The first CIC user who handled the conversation-based email.

First Workgroup Queue

The first workgroup who handled the conversation-based email.

Exit Paths

Success

The operation was successful.

Failure

The operation was not successful.

Query ConversationID Last Agent

This System tool returns the last agent and workgroup who handled the email conversation.

Note: This tool uses Interaction Tracker data, so you must have the Interaction Tracker Access or Recorder license for the tool to work. If the license is not available, the tool returns empty values.

Inputs

Email Conversation ID

The identifier for the email conversation for which you want to find the last user and workgroup.

Outputs

Last ICUser

The last CIC user who handled the conversation-based email.

Last Workgroup Queue

The last workgroup who handled the conversation-based email.

Exit Paths

Success

The operation was successful.

Failure

The operation was not successful.

Query License Details

This System tool takes the name of a counted license, such as the number of workstations or advanced scripster clients, and determines the number of licenses currently available.

Inputs

License Name

The feature being queried.

Outputs

Total Number of Licenses

The total number of licenses for this feature. If the license does not exist on the server, this number will be zero.

Currently Available

The number of licenses currently available for this feature. If the number does not exist on the server, this number will be zero.

Exit Paths

Licensed

This path is taken if the feature is licensed.

Not Licensed

This path is taken if the feature is not licensed. Note that this path will not be taken if a feature is licensed but has none available.

Query Security Policy

This System tool queries the currently configured password security policy parameters that are configured in Interaction Administrator. For more information, refer to Interaction Administrator help.

Inputs

User ID

The user for whom the handler is requesting security information.

Outputs

Number of passwords kept in your password history

Number of passwords that will be checked in the user's history (i.e., the number of unique passwords required before one may be reused). This corresponds to the "Minimum number of unique passwords before one can be reused" setting in Interaction Administrator.

Minimum time before you can change your password

This is the minimum amount of time before which a user can change his or her password. This requirement is set in the "minimum age of password before user can change it (days)" parameter in Interaction Administrator.

Minimum password length

The minimum number of characters that passwords must contain.

Number of "unique" digits required

The minimum number of digits, ignoring repeats, that the password must contain after DTMF conversion. For example, "abc" is "222" after DTMF conversion, which is only one "unique" digit. "Abcd," however, converts to "2223," so this is two unique digits.

Sequential Digits Allowed

Whether or not the entire password may be composed of sequential digits after DTMF conversion. For example, "123123" would not be considered "sequential" by this parameter, as the entire password is not sequential. However "123456" would be considered sequential.

Lockout Duration (minutes)

This is the number of minutes the account will be unavailable once the account has been locked out due to invalid login attempts.

Lockout reset time

The amount of time each invalid login attempt is remembered and counted against the total number of allowable attempts.

Minimum uppercase characters

The minimum number of uppercase characters that the password must contain.

Minimum lowercase characters

The minimum number of lowercase characters that the password must contain.

Minimum numeric characters

The minimum number of numeric characters that the password must contain.

Minimum special characters

The minimum number of special characters that the password must contain.

Special characters

The special characters that the password can contain to meet the requirement for minimum number of special characters. A password can include any of these special characters in any combination to meet the requirement. A password can include other special characters but the other special characters do not count toward the requirement.

Special characters can include: ~!@#\$%^&* _+ = `| \ () { } [] ; : " ' < > , . ? /

Exit Paths

Next

This tool always takes the Next exit path.

Quick Directory Available Check

The System tool is designed to quickly determine if a directory is available on the network. It takes three parameters: the directory name (UNC path), timeout for the check, and cache entry timeout

Note: The tool **must not** be used to check for availability of files, or large numbers of directories. It is for root share availability check **only**.

Inputs

Path Name

The UNC path of the directory being checked.

Request Timeout

The number of seconds the tool will wait for a return before timing out.

Cache Entry Refresh

The cache entry timeout specifies the amount of time last check will be cached. After that timeout expires, the system will automatically check whether the state of directory availability has changed from the state recorded in the cache.

Exit Paths

Success

This path is taken if the specified directory is available.

Failure

This path is taken if the specified directory is not available.

Timeout

This path is taken if the specified directory is not found in the allotted time.

RegEx Extended

This System tool allows the parsing of a string based on a standard regular expression. You might use this tool to check and match IP, e-mail, or SIP addresses within a string. Or, you might use it to extract information, such as a call ID or phone number, from an e-mail message.

Inputs

String

The string to parse.

Expression

A string containing the regular expression with which the string is to be parsed. The regular expression uses the following escape sequences:

"*"	Represents the zero or more repetition operator.
"+"	Represents the one or more repetition operator.
"?"	Represents the zero or one repetition operator.
"[" and "]"	Used to specify character ranges (sets). Use a dash to specify a range, such as "[a-z]".
"[^" and "]"	Used to specify character ranges not to match. For example, "[^0-9]" would match a non-number.
"{" and "}"	Used for bounded repetitions.
"(" and ")"	Used to group sub-expressions. You can also use this to specify captures.
" "	Represents the alternation operator.

The expression syntax supported in this tool uses the `regex_search` algorithm with `match_continuous`. This combination only matches a string sequence where the first character matches the provided regular expression. For more information, see <http://www.boost.org>.

So for example, to find an IP address in a String, the expression might look like this:

```
"([0-9]{1,3}+\.[0-9]{1,3}+\.[0-9]{1,3}+\.[0-9]+)"
```

Note: To get a single slash in the actual string at runtime, you need to enter double slashes. For example, to get the string `"\"` you need to enter `"\"` in the Interaction Designer editor.

Format

The format is used to specify the output of the regular expression. It is very important that at least one capture is specified in the regular expressions. A capture is specified using "(" and)". The first capture is represented as "\$1", the second as "\$2", and so on.

Case sensitive

When this check box is selected, case-insensitive comparisons are used. Clear this check box if you do not want the comparisons to be case-sensitive.

Repeat

It is possible to do a partial match of the input string. If there is a possibility for multiple matches, this option will continue searching until the input string is consumed, or a match does not consume anything.

Multi-Output

If a single match should return multiple outputs, then this option should be checked. When used, the first character in the format string is used as the output delimiter. For example, if the regular expression captured two pieces of information that needed to be output as two separate items, then the format would be `"$1;$2"`. The first character indicates that the `;` delimits the outputs. In this case, `"$1"` and `"$2"` are the separate outputs.

Outputs

Result

A string list containing the results of the expression applied to the parsed string.

Remainder

Because a partial match is possible, this string returns any input string that was not matched.

Exit Paths

Full Match

This path is taken if the input string is fully matched.

Partial Match

This path is taken if matches were found, but the input was not fully matched.

No Match

This path is taken if no matches were found.

Failure

This path is taken if there is an error in the regular expression. An example of an error would be the expression "one|" because the `|` operator is used without specifying the alternative.

Examples

The following example shows how to find all of the call IDs and phone numbers from an e-mail message.

String:

```
13:43:59: Initializing
13:43:59: Offering
13:44:06: Voice Mail
13:44:27: Disconnected [Remote Disconnect]
13:44:27: Caller has recorded and sent a Voicemail message
6 seconds
[ID: 2000061016#3173452098]
```

```
13:43:59: Initializing
13:43:59: Offering
13:44:06: Voice Mail
13:44:27: Disconnected [Remote Disconnect]
13:44:27: Caller has recorded and sent a Voicemail message
6 seconds
[ID: 2013061017#3173451234]
```

Expression:

```
.*\[ID:\[ \t]*([0-9]+)#([0-9]+)\]
```

Format:

```
;ID=$1;Number=$2
```

Repeat:

On

Multi-Output:

On

Retrieve External Password

This System tool is used to get login information from an external service (such as a mainframe).

Warning: It is possible for users with the ability to publish and debug handlers to view login information with this tool.

Inputs

Password Id

The unique identifier for this set of login information.

Outputs

Server Name

The server receiving the login information.

Account Name

The account for which the login information is needed.

Password

String containing the password information

Custom String Value

This parameter contains whatever additional information may be needed to log in.

Exit Paths

Success

This path is taken if the password is successfully retrieved.

Failure

This path is taken if the operation fails. This will happen if this tool is used without having previously run a corresponding [Put External Password](#) tool.

Semaphore Lock

This System tool attempts to acquire a lock on the resource semaphore named in the resource name parameter.

If the resource has not been initialized, the first use of a semaphore lock tool initializes the semaphore and the Shared Lock Limit.

Note: Locks are released when the handler placing them finishes. They can not be locked by one handler and released by another. They can be locked by subroutines and kept alive by passing Semaphore Lock Handle to the subroutine, and keeping the returned handle in scope until the calling handler completes.

Multiple calls to lock a resource by a handler and its subroutines will not place any additional locks. Instead a reference count is incremented. The reference count will be used to insure that the resource is not unlocked until a matching number of Unlocks is called.

Regardless of the number of times the lock is placed on the resource by the handler, when the handler exits, the resource lock is freed for another handler to access.

Inputs

Semaphore Resource Name

Identifies the resource to be locked.

Note: A named semaphore is not used. The name is used only to lookup and identify the correct semaphore object in memory. This is not a named semaphore that can be shared with other NT processes.

Shared Access Limit

Number of locks that can be placed against the semaphore at the same time. When this limit is reached, all others must wait until a lock is released. The default limit is 1.

Wait Timeout Milliseconds

Number of milliseconds the lock tool will wait for a lock before giving up and taking the timeout path. The default timeout is 500 milliseconds.

Outputs

Semaphore Lock Handle

This is the handle returned when a lock is successful. This handle must be used with the [Semaphore Unlock](#) tool. The handle is also part of the mechanism that insures that a lock will be released when the handler exits.

Exit Paths:

Success

Lock was given on resource.

Failure

Some type of internal error kept the locking process from succeeding. Trace logs should be checked for error messages.

Timeout

Lock was not acquired within the specified timeout limit.

Semaphore Unlock

This System tool releases the lock set by a [Semaphore Lock](#) tool. A valid Semaphore Lock Handle returned by the Lock tool must be used to perform the unlock.

An error will occur if more unlocks are called than locks that were placed by a handler.

Inputs

Semaphore Resource Name

Name of resource for which a lock should be released.

Semaphore Lock Handle

Lock handle returned by a Semaphore Lock tool in this handler, or returned by a subroutine.

Exit Paths:

Success

This path is taken if the resource name and Lock Handle were valid, and the lock is released. Another handler that has been waiting will immediately access this lock.

Failure

This path is taken if the resource name or Lock Handle was invalid, or a system error occurred. The trace logs can be used to determine why a failure has occurred.

Send Custom Notification

This System tool creates an event that starts the [Custom Notification initiator](#). You can pass information along with the event.

Inputs

Custom Object Identifier

The identifier for the custom object. This must match the Object ID in the Custom Notification initiator to run when this step executes.

Note: Because custom object identifiers remain in memory indefinitely, you should not use dynamically-generated strings in this field. Dynamically-generated strings will cause memory growth in the NotifierU.exe process.

Custom Event Identifier

The identifier for the event that starts a Custom Notification initiator. This must match the Notification Event in the Custom Notification initiator to run when this step executes.

Note: Because custom object identifiers remain in memory indefinitely, you should not use dynamically-generated strings in this field. Dynamically-generated strings will cause memory growth in the NotifierU.exe process.

Custom Notification Data

Any data to pass in with the notification event. The Custom Notification initiator captures this information and passes it into a handler. This parameter expects a list of strings value.

Exit Paths

Next

This step always takes the Next exit path.

Server Name

This System tool retrieves the host name of the CIC server.

Output

Server Name

A string containing the CIC server name.

Exit Paths

Next

This path is taken once the server name is retrieved.

Sleep

This System tool pauses a handler's execution for a specific number of seconds.

Inputs

Time to Sleep (seconds)

The number of seconds to pause the handler before continuing. Setting this value to 0 results in this handler thread giving up its timeslice so that NT executes other tasks that are waiting for CPU time.

Since this parameter accepts only integer values, decimal values are not allowed. To indicate a partial second, you can specify a negative value. Any negative values are interpreted as milliseconds (for example, -1500 is interpreted as 1.5 seconds and -3500 resolves to 3.5 seconds).

Caution: Setting large Time to Sleep values can cause handler threads to accumulate in Interaction Processor. We recommend that you use the Timer initiator.

Exit Paths

Next

This tool always takes the next exit path.

Tracker User Audit Data

This System tool is reserved for future use and is not currently implemented.

Tracker VoiceMail Data

This System tool is used to track voice mail or fax actions and their dispositions.

Inputs

Call Id Key

Call ID of the call taking action or leaving a voice mail (optional),

User Id

User name value from CIC.

User Organization

(Not used for CIC.)

VoiceMail Id

Guid created and assigned to the call or fax on creation, forward, or reply actions. If not supplied, operations cannot be chained to original vociemails or faxes. (Optional)

Duration

Message length on new, forward, or reply. (Optional for listen or delete action.)

Mailbox Extension

Targeted mailbox extension.

Action Start Time

Starting time of voicemail or fax.

Note: The difference between the start and end times may differ from the message duration because of other operations that can occur to speed up or slow down playback, recording, or processing.

Action End Time

Ending time of voice mail or fax. See note above.

Disposition Type

One of the following:

- (N)ew
- (L)isten
- (D)elete
- (F)orward
- (R)eply

Source Type

Source of the voicemail:

- a. ExternalTUI
- b. InternalTUI
- c. Application/Web
- d. External User
- e. Internal User
- f. Unknown

Exit Paths

Next

This tool always takes the Next exit path.

Transcribe Recording Begin

This System tool uses HTTPS (SSL) to securely send voicemail over the Internet to a service provider, such as Yap, who will attempt to accurately decode the message using natural speech processing. The resulting transcription is then returned to be inserted into the voicemail message using the [Transcribe Recording Result](#) tool.

Note: Voicemail transcription is a separately licensed add-on feature.

Inputs

Request URL

The URL provided by the transcription service provider, typically something like:

Request Options

An optional parameter used to provide the transcription service with optional request values. This parameter does not normally need to be set and handlers use a null value by default.

Note: This parameter is for future use.

Certificate

A certificate of authenticity provided by the service provider and consisting of a string of hexadecimal characters.

User ID

Identifies individual users or accounts for billing purposes. If a billing system is being used, this parameter is required for each transaction.

Speaker ID

Identifies the speaker and is used to track model adaptations for unique speakers to improve the accuracy of transcriptions. Use this parameter only if speakers can be reliably differentiated.

Note: You can have multiple, unique Speaker ID values for a particular User ID.

Filter Set

Provides the service provider with a context for the type of message. This parameter does not normally need to be set, but depends on the service provider and scenario. Your service provider might require values that are different from the ones listed below.

Value	Comment	Removes Pauses?	Replaces spelled numbers with digits?	Inserts limited punctuation?	Inserts commands and periods?	Replaces profanities with asterisks?
"none"	The default value if no filter set is specified. Results are returned with no presentation reformatting.	No	No	No	No	No
vm3 (voicemail 3)		Yes	Yes	Yes	No	Yes
vm4 (voicemail 4)	Recommended for voicemail to text applications. This is the default value if an empty string is used.	Yes	Yes	No	Yes	Yes
ff2 (freeform 2)		Yes	Yes	No	No	Yes

Note: If no value (an empty string) is passed, the tool uses the vm4 filter set.

Recording ID

The unique ID assigned to the recording.

Recording File

The name of the file that contains the recording.

Note: If both the recording ID and Recording File parameters are specified, the tool uses Recording ID.

Detailed Results

Attribute (True or False) that turns on/off detailed results from the transcription request. Handlers default to **False**. When set to **True**, detailed results will appear in the voicemail message body of the transcription. A value other than **True** or **False** is interpreted as **False**.

Timeout (sec)

Set to 45 seconds as the default, this value is used as a threshold for abandoning a transcription request. A voicemail is sent only after a transcription request has returned or the abandon threshold has been reached. The delay is added to any delay in voicemail delivery that may have existed prior to the activation of this feature. This value can be modified to meet specific needs, but keep in mind that using a smaller timeout value increases chances of a request being abandoned due to timeout.

Outputs

Transcription Cookie

A code identifying the transcription job as a referenced, counted handler object.

Note: The job is discarded if the object goes out of scope.

Transcription Cookie String

A string that refers to the object.

Exit Code

An Exit Code (iExitCode) of 0 (zero) indicates success and a non-zero code indicates an error or other condition about the results of this tool step.

Exit Text

The exit text string (sExitText) corresponds to the Exit Code. When the Exit Code is 0, the Exit Text is "Success."

Exit Paths

Success

This step takes the Success exit path if the voicemail was successfully sent to the transcription service.

Failure

This step takes the Failure exit path if the voicemail was not successfully sent.

This System tool is used in conjunction with the [Transcribe Recording Begin](#) tool to return the transcription of the recorded voicemail message.

Inputs

Transcription Cookie

The reference counted handler object identifying the transcription job.

Note: The job is discarded if the object goes out of scope.

Transcription Cookie String

A string that refers to the handler object.

Note: One of the two parameters above (Transcription Cookie or Transcription Cookie String) is required. If both are provided, the Transcription Cookie has preference. The string is a convenience because passing the object around can be difficult in some cases.

Timeout (sec)

Set to 45 seconds as the default, this value is used as the amount of time to wait for the transcription result.

Note: This timeout is the amount of time to wait to see if a result is available. It is not the same as the timeout of the Transcribe Recording Begin tool, which specifies the amount of time to wait for the transcription to be received from the service provider. In most cases, it is sufficient to set the two timeouts to the same value if the asynchronous capabilities of the tool pair are not required.

Outputs

Transcription

The transcribed text of the submitted audio file.

Transcription XML

The XML-tagged version of the transcribed text.

Exit Code

An Exit Code (iExitCode) of 0 (zero) indicates success and a non-zero code indicates an error or other condition about the results of this tool step.

Exit Text

The exit text string (sExitText) corresponds to the Exit Code. When the Exit Code is 0, the Exit Text is "Success."

Exit Paths

Success

This step takes the Success exit path if the result was received successfully.

Timeout

The result was not acquired within the specified timeout limit.

Failure

This step takes the Failure exit path if the result was not successfully received.

Unique ID

This System tool returns a string identifier which is unique over a relatively long period of time (a little over 31 years). The string returned is always composed of twelve decimal digits. When this tool is first called, the identifier is seeded with the number of milliseconds since 1/1/1999 and its value is incremented by one on each successive call.

The tool was developed for use by Interaction Director, but should also be useful as a unique key for database operations.

Outputs

Unique String Identifier

The unique twelve decimal digit string.

Exit Paths

Next

This tool always takes the next exit path.

Verisign Transaction

This System tool, when used with the PayFlow Pro application from PayPal, allows the use of PayPal/Verisign credit card transactions. For information, refer to the *PayFlow Pro* documentation available from PayPal (formerly Verisign).

Exit Paths

Success

This path is taken if the transaction is successful.

Failure

This path is taken if the operation fails.

Whitepages

This System tool performs a synchronous Reverse Whitepages (RWP) lookup, which means processing is suspended while the lookup is performed. Behind the scenes, this tool initiates a request that instructs Data Manager to perform a RWP lookup against the data sources that are configured to participate. It waits for the search to complete, and then returns the results.

Typically, this tool is used to populate the body of a voice mail message with directory information or to populate the Name field for a call in a queue. (See the note below about how the EIC_RemoteName call attribute is populated.) This tool compares a telephone number to the phone numbers listed in the WhitePages.txt file and returns any associated directory information listed there. The WhitePages.txt file is located in the Resources directory on your CIC server (for example, <CIC Installed Drive>\I3\IC\Resources). If Whitepages cannot find an associated name and the telephone number starts with a "1," it is looked up based on the area code and exchange for North America.

The WhitePages.txt file may be updated while the CIC system is running; changes to it will take effect shortly after the file is

updated. The file consists of one entry per line; each entry consists of a partial (or full) telephone number followed by white space followed by address information. Specify a full address by separating lines with the pipe (|) character (the pipe is replaced with a new line character when the directory information is returned from the tool).

The following sample entries are taken from the WhitePages.txt directory:

```
1-317-222-2222 Time & Temperature
```

```
1-317-872-3000 Interactive Intelligence|7601 Interactive Way|Indianapolis, IN 46268
```

```
20 Egypt
```

All lookups in WhitePages.txt are done on a longest-matching leading string basis, thus a number beginning with 6724 will match Christmas Island but a number beginning with 6725 will match Antarctica. Antarctica has an entry of 672; Christmas Island has an entry of 6724. There is no entry for 6725, so it gets matched with the longest match available (Antarctica).

Note: In its current form the Whitepages tool is not designed for extremely large numbers of entries in the WhitePages.txt file. We have experimented with extracting data from CD-ROM sources for use with the tool, but the licensing problems involved with doing so have precluded us from offering such a service. Since the CIC system needs to access the reverse white pages data in real time we do not support the use of databases from CD-ROMs nor do we support an interface to third party tools.

Note on how Caller ID is looked up: Whitepages.txt is not the first place the CIC looks when populating the Name field (which is the EIC_RemoteName call attribute) for a call in a queue. CIC attempts to populate this field from several data sources, in a specific order described below:

1. CIC uses the phone number/address information to do a white pages name lookup. If the lookup returns a value, CIC assigns that value to Eic_RemoteName.
2. Otherwise, if the CO delivered caller name, CIC assigns that value to Eic_RemoteName.
3. Otherwise, if a formatted remote number was returned by white pages lookup, CIC assigns it to Eic_RemoteName.
4. Otherwise, CIC assigns "Unknown."

The Whitepages tool performs a *synchronous* lookup, meaning that it waits for the search to complete and then returns the results. Processing is suspended while the search is performed.

Inputs

Interaction Address

The telephone number, fax number, email address, or web address to look up in the WhitePages.txt file. The type of address information must be specified in the following input field. If a telephone number is specified, the input must be a unique international number. This means that numbers for North America must include an area code and must start with "1"; international codes must start with the appropriate country code.

Interaction Address Type

Specifies the type of input contained in the Interaction Address input.

1=Phone number

2=Fax number

3=Email address

4=Web address

Interaction Direction

Specifies the direction of the interaction. If the search is not associated with an interaction, or is not direction-sensitive, specify "3" for Any.

1=Incoming

2=Outgoing

3=Any

Public/Private Sources

Specifies the source types to include in the search. Note that if 2 (Private) or 3 (Both) are specified, the CIC User ID must be specified in the following field.

1=Public

2=Private

3=Both

IC User ID

Use for private searches only. Specifies the CIC User ID to use for searches against private RWP sources.

Contact List Sources

Specifies which Data Manager Contact List sources to use when performing the RWP search.

Separate items with a plus sign (+). Use '*' for the default RWP sources defined in Interaction Administrator.

Result Format

Specifies the format of the search results string.

1=Display Name

2=Contact Summary

3=XML

Each of these formats is described below.

Format	Description
Display Name	A single display name, as generated by the CIC display name generation/formatting facility. If there are multiple matched rows, then only the data from the first row is used. An example display name is: John Q. Public (XYZ Corp)

Contact Summary	<p>A short summary of the contact data for the first matched row. This format is used for things like fax and voicemail headers, mailing addresses, etc. The first line is always the display name, as generated by the CIC display name generation/formatting facility. The next lines (if present) contain address information.</p> <p>For example:</p> <p>John Q. Public XYZ Corp. Finance 1 Oak St. Albany, NY 12345 USA</p> <p>If a primary phone, email, fax, chat, or web URL is non-empty, it is also added (and prefixed accordingly).</p> <p>An example if all are non-empty:</p> <p>John Q. Public XYZ Corp. Finance 1 Oak St. Albany, NY 12345 USA phone:(333) 444-5555 fax:(333) 444-5556 mailto:john.public@xyz.com chat:john.public@xyz.com http://www.xyx.com</p>
------------------------	--

XML	<p>An XML string. This is the only format that may contain more than one result row. It is also the only format that allows applications to access driver-specific attributes. This format is used internally for almost all RWP processing.</p> <p>An example containing two result rows:</p> <pre><?xml version="1.0"?> <rows> <row Individ="iovq*w~m&amp;du#~5* 090 v3" FirstName="John" LastName="Doe" Department="Marketing" IsPrivate="0" LocID="v9y+oec=&amp;gaz\$9f,[w^93" LocName="Acme-Toledo" OrgID="cbrwl4h.*&amp;#n9to\$0?@&amp;?3" OrgName="Acme, Inc." Phone="+12223334444^123" Email="john.doe@acme.com" InteractionAddress="+12223334444" InteractionType="1" Source="I3Tracker Public Rwp" ExtSource="I3Tracker Public Rwp" Replicate="0" IsStopSource="1" ICInteractionID="2100176223" ICInteractionAttribute="Eic_RemoteName" DisplayName="Doe, John (Acme, Inc.)" ContactSummary="Doe, John&#xA;Acme, Inc.&#xA;Marketing&#xA;phone: (222)333-4444 ext 123"&#xA; Origin="2" IsDefinitiveResult="0"/> <row Individ="e8x`.d#.d~;np6sxh&gt;" FirstName="Jane" LastName="Doe" Department="Sales" IsPrivate="0" LocID="{b>8b};u(>4v{#;388}4" LocName="Acme- Cleveland" OrgID=" cbrwl4h.*&amp;#n9to\$0?@&amp;?3" OrgName="Acme, Inc." Phone="+12223334444^124" Email="jane.doe@acme.com" InteractionAddress="+12223334444" InteractionType="1" Source="I3Tracker Public Rwp" ExtSource="I3Tracker Public Rwp" Replicate="0" IsStopSource="1" ICInteractionID="2100176223" ICInteractionAttribute="Eic_RemoteName" DisplayName="Doe, Jane (Acme, Inc.)" ContactSummary="Doe, Jane&#xA;Acme, Inc.&#xA;Sales&#xA;phone: (222)333-4444 ext 124"&#xA; Origin="2" IsDefinitiveResult="0"/> </rows></pre>
-----	--

Timeout

The time, in seconds, before this tool will return with a timeout error.

Send result(s) to Interaction Tracker

If TRUE, the results of the search will be recognized by Interaction Tracker (only if an Interaction Tracker license is found).

If FALSE, the results of the search are not sent to Interaction Tracker.

LocationFilter

Specifies a string that indicates a Location Filter specified in the Dial Plan. By default, the values can be "<All>", "<Default Location>", or "<Skip this item>" but there may be many custom Location Filters listed as well, if they were previously defined in the Regionalization configuration. Enter the Location Filter name as it appears in the Interaction Administrator dial plan.

Outputs

Directory Information

The results of the search. The format of this data depends on the value of the Result Format input parameter.

Formatted Interaction Address

A formatted version of the address information used in the search.

Exit Paths

Next

This step always takes the Next exit path.

WhitePagesAsync

This System tool sends an asynchronous RWP lookup request to Data Manager (DM) for an interaction to retrieve the caller's name. See the *Reverse Whitepages Lookup* white paper for a detailed explanation of how this tool is used with CIC.

Inputs

Interaction Address

The phone number, telephone number, email address, or web address to look up in the specified source(s).

Interaction Address Type

Specifies the type of information in the Interaction Address input field.

1=Phone number

2=Fax number

3=Email address

4=Web address

Interaction Direction

Specifies the direction of the interaction. If this lookup is not associated with an interaction or is not direction-sensitive, specify "3" (Any).

1=Incoming

2=Outgoing

3=Any

Public/Private Sources

Species the source types to include in the search. Note that if "2" (Private) or "3" (Both) are specified, the CIC User ID must be specified in the field below.

1=Public

2=Private

3=Both

IC User ID

Used for private searches only. Specifies the CIC User ID to use for searches against private RWP sources.

Call Identifier

The ID of the interaction whose specified attribute(s) will be set with the results.

Attribute Name

The name of the attribute(s) to set with the results. In general, only one attribute, EIC_RemoteName, is set; however, you can set multiple attributes by listing their names, separated by '+'.

Contact List Sources

Specifies which Data Manager Contact List sources to use when performing the RWP lookup.

Result Format

Specifies the format of the result(s) string, which is used to set the specified interaction attribute(s). For descriptions of the supported format options, see [Whitepages](#).

1=Display Name

2=Contact Summary

3=XML

Default Display Name

The display name to use if no match is found for the specified interaction address.

Send result(s) to Interaction Tracker

If True, the results of the search are recognized by Interaction Tracker (only if a valid Interaction Tracker license is found).

If False, the results are not sent to Interaction Tracker.

LocationFilter

Specifies a string that indicates a Location Filter specified in the Dial Plan. By default, the values can be "<All>", "<Default Location>", or "<Skip this item>" but there may be many custom Location Filters listed as well, if they were previously defined in the Regionalization configuration. Enter the Location Filter name as it appears in the Interaction Administrator dial plan.

Exit Paths

Next

This tool always takes the Next exit path.

WhitePagesLocality

This System tool retrieves the locality information (i.e., city, state, etc.) for an incoming call. See the *Reverse Whitepages Lookup* white paper for a detailed explanation of how this tool is used with CIC.

Inputs

Interaction Address

The telephone number, fax number, email address, or web address to be looked up.

Interaction Address Type

Specifies the type of information in the Interaction Address input field.

1=Phone number

2=Fax number

3=Email address

4=Web address

LocationFilter

Specifies a string that indicates a Location Filter specified in the Dial Plan. By default, the values can be "<All>", "<Default Location>", or "<Skip this item>" but there may be many custom Location Filters listed as well, if they were previously defined in the Regionalization configuration. Enter the Location Filter name as it appears in the Interaction Administrator dial plan.

Outputs

Locality Information

The city, state, and/or country information for the incoming call.

Formatted Interaction Address

A formatted version of the phone number/address information used in the search.

Exit Paths

Next

This tool always takes the Next exit path.

TUI

Attendant Email Logical Transfer

This TUI tool is for internal use only.

Create Calendar Session

This TUI tool creates a calendar session identifier suitable for use with the [Calendar tools](#) in the Calendar palette. The credentials and host for the session are taken from the Providers tab configured in the Mail container in Interaction Administrator.

Inputs

None

Outputs

Calendar Session

The calendar session identifier (idSession) to use as input for the [Calendar](#) tools.

Exit Code

An Exit Code of 0 (zero) indicates success and a non-zero code indicates an error or other condition about the results of this tool step.

Exit Text

The exit text string (sExitText) corresponds to the Exit Code. When the Exit Code is 0, the Exit Text is "Success."

Exit Paths

Success

The calendar session ID was successfully created.

Failure

See the Exit Code and Exit Text outputs for an explanation of any possible failure condition.

Create Private Contact Grammar

This TUI tool creates an ABNF grammar file containing the private contacts grammar according to the criteria specified as parameters. The grammar rule created by the tool has the following structure:

```
LeadingFiller Names TrailingFiller
```

The names are rendered depending on the various parameters. Assuming all options are enabled, the following rule fragment is rendered for each user:

```
[Jane] Doe | Jane [Doe] | Doe Jane.
```

Enabling the 'Last Name Optional' or 'First Name Optional' parameters will allow broader recognition (i.e. the callers don't have to say the full name). However, for large organizations this will lead to lots of ambiguous matches, in particular for common names such as "Smith".

NOTE: All selected slots are always returned, even if the caller doesn't say the corresponding component. For example, if the first name is optional and the user just says the last name of a person, the "firstname" slot will nevertheless be filled with the first name of the directory entry.

IMPORTANT: Do not use this tool to overwrite a preloaded grammar or a grammar that may be in use. Instead, a new grammar file with a unique name, for example by appending the current time and date, should be created. Rendering the private contacts list into the file may take quite some time. Overwriting a grammar file that may be used by the application could thus lead to a partially completed file being loaded and cause the grammar parser to fail. Creating a new file and only referencing it after it is completed prevents such problems. Use the [Reco Add Preloaded Grammar](#) tool to add/replace a preloaded grammar with the newly created private contact grammar.

Inputs

XML Document Handle

A XML document handle to be used to create the private contact grammar. Usually, this is the output of other voice recognition tools.

Grammar File Destination

A string variable of the full path of where the new grammar will be written.

Language

A string variable of the language to specify in the grammar.

If undefined, all users will be included.

Leading Filter

ABNF Rule fragment rendered as filler before the actual name tokens. For example:

```
[dial | call | (transfer | connect) [me] [to | with]]
```

NOTE: The filler must be in the SRGS ABNF format and is not rendered as optional. Thus, specifying "dial" as filler will mean that the user *must* say "dial" followed by the name for the grammar to match.

If undefined, no leading filler is rendered.

Trailing Filter

ABNF Rule fragment rendered as filler after the actual name tokens. For example:

```
[please]
```

NOTE: The filler must be in the SRGS ABNF format and is not rendered as optional.
If undefined, no trailing filler is rendered.

Custom Headers

Custom header fields and rules to inject into the SRGS ABNF grammar. The following headers are created by the tool and cannot be injected: 'mode', 'language', 'root'.

First Name Optional

This checkbox specifies whether the grammar should be rendered with an optional first name. If left unchecked, the caller must say the first name of the person. If checked, the first name of the user is optional and callers may just say the last name.

Last Name Optional

This checkbox specifies whether the grammar should be rendered with an optional last name. If left unchecked, the caller must say the last name. If checked, the last name of the user is optional in the grammar and callers may just say the first name.

Recognize "Last, First" as well?

This checkbox specifies whether user may say "last name, first name" in addition to "first name, last name". If left unchecked, the grammar expects "first name, last name". If checked, the user may say either "first name, last name" or "last name, first name".

Exit Paths

Success

The grammar file has been created successfully.

Failure

Creation of grammar file failed. Consult the Error Code parameter for details.

Find Localized File

This TUI tool retrieves the path to a localized version of an audio file (wav). Specify a path to the base file and then the tool retrieves the path to the version of the file based on the language you request. If a language is specified, it has precedence otherwise; the language attribute of the interaction is used. If no language attribute is set on the interaction, the default system language will be used. Both the InteractionID and the language are optional.

The name match is determined by the name of the file. For example, when searching for ringback.wav in English any of these files would be returned:

- Ringback.en-US.wav
- Ringback_en-US.wav
- Ringbacken-US.wav
- Ringback.en.wav
- Ringback_en.wav
- Ringbacken.wav
- Ringback.wav

Inputs

Interaction ID (optional)

An optional InteractionID that determines the language for the search. The language attribute on the interaction is used.

File Path (string)

A string of the full path for the base file, for example C:\I3\IC\Resources\Ringback.wav

Language (optional)

The language code of the prompt phrase to be played. If no language is specified, the language attribute of the interaction will be used. If no language is specified for the interaction either, the default language will be used.

Outputs

Path to localized file (string)

A string that contains the path to the localized version of the file, if it exists. For example, C:\I3\IC\Resources\Ringback-en_US.wav

Exit Paths

Success

The Success path is taken if a matching localized file was found.

Failure

The Failure path is taken if no matching localized file was found.

Find User By TN

This TUI tool provides the name of a user queue based on a phone number associated with that queue. When given a phone number, the tool searches for a user who has this phone number set as attribute value for (in order):

- Custom user attribute "Mobile Office ANI"
- Home Phone 1
- Home Phone 2
- Mobile Phone
- Business Phone 1
- Business Phone 2

The phone number may not be stored in the user's settings in the same format as the Phone Number parameter so the tool will try to use the following patterns (in order) for the search:

- The "Mobile Office ANI Pattern" server parameter
- XXXXXXXXXXX
- XXX-XXX-XXXX
- (XXX)XXX-XXXX
- (XXX) XXX-XXXX
- 1-XXX-XXX-XXXX
- 1-(XXX)-XXX-XXXX

Inputs

Phone Number (string)

A string value of a phone number to search for.

Outputs

Queue Identifier

The fully-scoped user queue found to match the search criteria. If more than one queue fits the criteria no results are returned and the Multiple exit path is taken.

Exit Paths

Found

A single matching user was found and the fully-scoped user queue has been returned.

Multiple

More than one user has a matching phone number so no Queue Identifier is returned.

Not Found

No users matched the phone number entered.

Get File Age

This TUI tool is for internal use only.

TUI Compile from String

This TUI tool is for internal use only.

TUI Get Catch

This TUI tool is for internal use only.

TUI Get Menu Attributes

This TUI tool is for internal use only.

TUI Session

This TUI tool is for internal use only.

Telephony

Telephony

The Telephony tools perform a variety of call-related functions within CIC. Click one of the tools below for more information on that tool.

[Add Party](#)

[Alert](#)

[Alert Station Group](#)

[Alert Workgroup](#)

[Answer](#)

[Assemble Prompt Phrase](#)

[Assemble String From Attributes](#)

[Assemble Text Phrase](#)

[Auto Conference](#)

[Bind Provisional Station](#)

[Blind Transfer](#)

[Change Workgroup Queue Initiator](#)

[Complete Parallel Make Call](#)

[Compress Audio File](#)

[Conference](#)

[Convert Call Id to String](#)

[Convert Conference ID to String](#)

[Convert String to Call Id](#)

[Convert String to Conference ID](#)

[Deferred Answer](#)

[Describe Phone Number](#)

[Disconnect](#)

[End TDD](#)
[Extended Blind Transfer](#)
[Extended Get Key](#)
[Extended Get Key Async](#)
[Extended Place Call](#)
[Flush Audio](#)
[Flush Keys](#)
[Get Attribute](#)
[Get Attributes](#)
[Get Billing Rate](#)
[Get Call Log](#)
[Get Datetime Attribute](#)
[Get Datetime Attributes](#)
[Get Key](#)
[Get Station Info](#)
[Get User's Location](#)
[Get Users with Role](#)
[Get Wildcard Attributes](#)
[Hold](#)
[Is Alertable?](#)
[Key Word Spotting](#)
[Listen From Call](#)
[Listen From Station](#)
[Log Message](#)
[Malicious Call Trace](#)
[Mute](#)
[Parallel Make Call](#)
[Parallel Make Call Failure](#)
[Park](#)
[PickUp](#)
[Place Call](#)
[Play Audio File](#)
[Play Digits](#)
[Play Prompt](#)
[Play Prompt Extended](#)
[Play Prompt Phrase](#)
[Play Recording](#)
[Play String](#)
[Play String Extended](#)
[Play Text File](#)
[Play Text File Extended](#)
[Play Tone](#)
[Priority Set Attributes](#)

[Private](#)
[Query Conference Properties](#)
[Query Logged In Users](#)
[Query Media Type](#)
[Query Monitored Queues](#)
[Query Queue](#)
[Queue Query Type](#)
[Query Statuses for User](#)
[Query User Status](#)
[Record Audio](#)
[Record Call](#)
[Record File](#)
[Record String](#)
[Record String Extended](#)
[Record Text File](#)
[Record Text File Extended](#)
[Reload Station](#)
[Remove Party](#)
[Reset Password](#)
[Secure Session Begin](#)
[Secure Session End](#)
[Secure Session Get Key](#)
[Secure Session Info Insert](#)
[Secure Session Info Validate](#)
[Select Call](#)
[Send ADSI String](#)
[Set Attribute](#)
[Set Attributes](#)
[Set Billing Rate](#)
[Set Call State](#)
[Set Datetime Attribute](#)
[Set Datetime Attributes](#)
[Set DTMF Password](#)
[Set State String](#)
[Set User Status](#)
[Set Visual Indicator](#)
[Station Answer](#)
[Station Audio](#)
[Station Connection Confirmation](#)
[Station Group Pickup](#)
[Station Pickup](#)
[Station Place Call](#)
[Synchronous Answer](#)

[System Queue](#)

[Transfer](#)

[Unbind Station](#)

[Unhold Call](#)

[User Login List](#)

[Validate DTMF Password](#)

[Verify Conference ID](#)

[Verify Interaction ID](#)

[Wait For Call On Queue](#)

[Wait for Disconnect](#)

[Wait for Monitor End](#)

[Wait Wrap Up](#)

[Wink](#)

[Zone Page](#)

Related Topics

[.symbol](#)

[/symbol](#)

Add Party

This Telephony tool adds a person connected on a line to a conference call.

Note: A conference call may contain no more than 32 calls.

Inputs

Call Identifier

The unique identifier for a call.

Conference Call Identifier

The identifier for the conference call you want to add a party to.

Exit Paths

Success

This step takes the Success exit path if the Conference ID and the Call ID are valid.

Failure

This tool can take the Failure exit path for several reasons. Failure can occur if the call disconnects, if the call is picked up by a user (and is no longer under the control of the handler), the call ID or conference ID is no longer valid (if the call is deallocated), or system resource limitation.

Alert

This Telephony tool notifies a station queue of an interaction that needs to be picked up. Alert rings any station the interaction's recipient is logged into. If the interaction's recipient is a workgroup or user queue, any recipient monitoring that queue from a CIC

client is alerted to the interaction.

Note: The Alert tool generally supports all CIC interaction types - calls, chats, email, generic objects, SMS, etc. - but not all of the Input parameters apply to each type of interaction, except calls. All of the parameters can apply to calls.

All non-call interactions can be affected by the Timeout, ACD Call?, and ACD User ID parameters. In addition, email and generic object interactions can use the Auto-answer parameter.

Alert times out if the interaction is not answered within a specific time period. This tool also accepts a caller's escape key inputs, and stores any escape keys entered for further IVR processing. For example, if a caller presses the "0" key while the Alert tool is alerting, the Alert tool takes the Escape exit path.

Alert changes the state of an interaction to Alerting. Only interactions in a state of Offering, On Hold, or Voice Mail can be acted upon by the Alert tool.

This tool also cancels any pending operations when transferring an ACD call to a user's queue.

Note on whisper: When a call alerts on a user's queue, you can choose to play a tone, a .WAV file, or both to inform the agent about the incoming call. This is useful if you are using auto-answer but want to warn the agent that another call is about to connect to their queue. The agent must be configured for auto-answer in the user properties in Interaction Administrator.

Inputs

Call Identifier

The unique identifier for an interaction that will alert.

Station Queue Identifier

The identifier for the station that receives the alerting call or communication.

Escape Keys

The valid keys a caller can press for this tool to take the Escape exit path.

Audio File Name

The name of the audio file (.WAV) played for the caller while the call is alerting. This might be music or a message describing the valid escape keys. If you specify a path, make sure it is a fully qualified path indicating server or drive letter. If you do not type a path, this tool uses the path stored in the Resource Path server parameter. MP3 files are not supported.

Timeout (seconds)

The amount of time this step alerts the station before taking the Timeout exit path.

ACD Call?

Set this value to true if this is an ACD interaction. Set this value to false if this is not an ACD interaction.

ACD User ID

The user ID of the agent to receive this interaction if this call is an ACD interaction.

Auto-answer if station is off-hook?

Set this value to true to have the call, generic object, or e-mail interaction placed on the queue in a state of "Connected" if the station is off-hook. This is useful for agents who are taking calls on a stand-alone phone station.

If the interaction type is e-mail and this value is "true," then the e-mail is immediately connected without the agent using the Pickup command.

Calling Number to be displayed

This string value can be displayed on an ADSI phone or an analog phone with a display.

Calling Name to be displayed

This string value can be displayed on an ADSI phone or an analog phone with a display.

Ring Cadence

Determines the ringing cadence used to alert the called party. The following table describes the values you can specify and the rings they produce:

Value	Ring produced: Ring On, (Ring Off)
1 (default)	2 seconds on, (4 seconds off)
2	1 seconds on, (5 seconds off)
3	.5 seconds on, (5.5 seconds off)
4	.5 seconds on, (.25 seconds off), .5 seconds on, (4.75 seconds off)
5	.5 seconds on, (.25 seconds off), .5 seconds on, (.25 seconds off), .5 seconds on, (4 seconds off)
6	.25 seconds on, (.25 seconds off), .25 seconds on, (.25 seconds off), .25 seconds on, (.25 seconds off), .25 seconds on, (4.25 seconds off)
7	1.25 seconds on, (.25 seconds off), .5 seconds on, (4 seconds off)
8	.5 seconds on, (.25 seconds off), 1.25 seconds on, (4 seconds off)

Note: When ringing a station with a phone type configured as Caller ID or ADSI (i.e., a phone that can display caller ID), TS will always use the default ring cadence (2 seconds on, 4 seconds off), regardless of which cadence is specified.

Whisper Tone Frequency

The tone frequency to be played to the agent receiving the whisper. Setting whisper tone frequency to 0 turns off whisper.

Whisper Tone Amplitude

The volume of the tone. Specify a higher number here to make the tone louder.

Whisper Tone Duration

The number of seconds the tone plays. Setting whisper tone duration to 0 turns off whisper.

Whisper File Name

The name of a .wav file to play. If you specify a path, make sure it is a fully qualified path indicating server or drive letter. If you do not type a path, this tool uses the path stored in the Resource Path server parameter. The .WAV file is played entirely after the tone. If tone or frequency are set to 0, the .WAV file will still play. .MP3 files are not supported.

Frequency 1 in Hertz

The frequency of the tone you want to generate. Set to 0 to disable call waiting.

Frequency 1 Amplitude in dB:

The amplitude of the frequency you want to generate. By default this is set to -10.

Frequency 2 in Hertz:

The frequency of the second tone you want to generate. By default this is set to 440 Hz.

Frequency 2 Amplitude in dB:

The amplitude of the frequency you want to generate. By default this is set to -10.

Call Waiting Tone Duration

The number of seconds the tone plays. To convert milliseconds to seconds, divide your millisecond value by 1000. For example, if you want the tone to play for 500 milliseconds, you would type .5 in this parameter.

Silence Duration After Call Waiting Tone

The number of seconds of silence before the call waiting tone repeats.

Call Waiting Tone Repeat Count

The number of times the call waiting tone plays before the call is transferred to voice mail.

Agent Greeting .WAV File

The name of a .wav file to play for the agent greeting. If you specify a path, make sure it is a fully qualified path indicating server or drive letter. If you do not type a path, this tool uses the path stored in the Resource Path server parameter. The .WAV file is played entirely after the tone. If tone or frequency are set to 0, the .WAV file will still play. .MP3 files are not supported.

Alert Primaries?

Primaries, in this case, refers to primary station appearances, such as SIP phones that support shared line appearances. The primary phone is the one a call is targeted at (e.g., 7113) and the secondary or shared line appearances are the buttons or lights on other phones that might monitor the same line. You do not need to set this parameter as the Telephony Services system sets it automatically based on the user status. If the target user's status is "In a meeting", Alert Primaries? is set to False, as it should not ring the target user's station, but it may ring any secondary line appearances for that station.

Monitoring Stations

List of stations that are monitoring the specified queue.

Monitoring Users

List of users who are monitoring the specified queue.

SIP Ring Type

One of the following:

- Internal
- External

- DID

Outputs

Keys

The escape digit(s) pressed by the caller if the caller tries to cancel or speak with an operator while the call is alerting.

Exit Paths

Answered

The call was answered by the intended recipient. A handler can no longer act on the call object while the call's recipient controls the call object. (Call recording will continue even if the handler no longer has control of the call.)

Escape

The caller pressed an escape key.

Timeout

The call was not picked up.

Failure

This tool can take the Failure exit path for several reasons. Failure can occur if the call disconnects, if the call is picked up by a user (and is no longer under the control of the handler), the call ID is no longer valid (if the call is deallocated), or system resource limitation.

Alert Station Group

This Telephony tool alerts an entire station group to an incoming call. Every agent with an available ACD status that is a member of the specified station group will be alerted. This tool is highly configurable with different options for how the station group is alerted. Alert Station Group allows you to set up a station group as a linear hunt group or a round robin hunt group.

Inputs

Call Identifier

The identifier for the call to alert on the workgroup.

Station Group

A unique identifier for the station group to alert.

Escape Keys

The keys a caller can press to escape the alert mode and take the escape exit path.

Audio File Name

The .wav file played to callers while the call alerts. Common uses are music, keypress instructions, or both. If you specify a path, make sure it is a fully qualified path indicating server or drive letter. If you do not type a path, this tool uses the path stored in the Resource Path server parameter.

Timeout

The number of seconds to alert before taking the Timeout exit path for special processing.

Alert Users Simultaneously?

Set this parameter to True to alert every user in the workgroup at the same time. Set to False to alert users sequentially (one at a time) using the order configured for that workgroup in Interaction Administrator. See the Maintain Order option in the Interaction Administrator workgroup configuration container and the *ACD Processing* technical reference (located in your CIC Server documentation directory on the CIC server) for more information on configuring these linear hunt groups.

Linear hunt groups always alert the first workgroup member as listed in Interaction Administrator. If that user does not pick up, or if the call could not be assigned to that user, then the next user in the list is alerted. This means that users at the end of the list will receive calls less often.

Linear is not used if Round Robin Alerting is selected.

Round Robin Alerting?

Round robin alerting is similar to linear hunt groups, but CIC remembers which members have received calls and alerts the person who has not received a call for the longest amount of time. In Interaction Administrator, you can select the Maintain Order option for a list of workgroup members. If you set this to True, the order of the members is used in deciding which user to alert. This option is ignored if Alert Users Simultaneously is set to True.

For example, a workgroup has five users (User1 - User5), all available for workgroup calls. User1 is the first member of this workgroup as configured in Interaction Administrator. User2 is member 2, and so on.

If Alert Station Group is invoked on call #1 and shortly thereafter on call #2, call #1 would alert on User1 and call #2 would alert on User2. If neither answers, call #1 will roll over to User3 and call #2 will roll over to User4. If User4 answers call #2 but User3 doesn't answer call #1, call #1 will roll over to User5. If call #1 is still not answered, it will roll over to User2. If there is still no answer, call #1 will roll over to User4 if User4 has finished with call #2, and will take the Timeout exit if not.

If at some later time another call invokes Alert Station Group, the first user considered is not necessarily User1, but is the "next" user. That is, if User2 answered call #1 in the above scenario, the "next" user to try would be User3.

Calling Number to Be Displayed?

Used to display information on ADSI phone or some other feature phone.

Calling Name to be displayed?

Used to display information on ADSI phone or some other feature phone.

Frequency 1 in Hertz

The frequency of the tone you want to generate. By default this is set to 1100 Hz.

Frequency 1 Amplitude in dB:

The amplitude of the frequency you want to generate. By default this is set to -10.

Frequency 2 in Hertz:

The frequency of the second tone you want to generate. By default this is set to 0 Hz.

Frequency 2 Amplitude in dB:

The amplitude of the frequency you want to generate. By default this is set to -10.

Call Waiting Tone Duration

The number of seconds the tone plays. To convert milliseconds to seconds, divide your millisecond value by 1000. For example, if you want the tone to play for 500 milliseconds, you would type .5 in this parameter.

Silence Duration After Call Waiting Tone

The number of seconds of silence before the call waiting tone repeats.

Call Waiting Tone Repeat Count

The number of times the call waiting tone plays before the call is transferred to voice mail.

Outputs

Keys

Contains any keys the caller pressed.

Exit Paths

Answered

This tool takes the answered exit path if the call was picked up by a member of the workgroup. The handler no longer has control of the call after that.

Escape

This tool takes the Escape exit path if the caller presses an escape key.

Timeout

This tool takes the Timeout exit path if the call alerts longer than the number of seconds specified in the Timeout parameter.

Failure

This tool can take the Failure exit path for several reasons. Failure can occur if the call disconnects, if the call is picked up by a user (and is no longer under the control of the handler), the call ID is no longer valid (if the call is deallocated), or system resource limitation.

No Stations Alertable

This path is taken if there are no stations alertable on the server.

Alert Workgroup

This Telephony tool alerts an entire workgroup to an incoming call. Every agent with an available ACD status that is a member of the workgroup to which the interaction has been assigned will be alerted. This tool is highly configurable with different options for how the workgroup is alerted. Alert Workgroup allows you to set up a workgroup as a linear hunt group or a round robin hunt group.

Inputs

Call Identifier

The identifier for the call to alert on the workgroup.

Escape Keys

The keys a caller can press to escape the alert mode and take the escape exit path.

Audio File Name

The .wav file played to callers while the call alerts. Common uses are music, keypress instructions, or both. If you specify a path, make sure it is a fully qualified path indicating server or drive letter. If you do not type a path, this tool uses the path stored in the Resource Path server parameter.

Timeout

The number of seconds to alert before taking the Timeout exit path for special processing.

Alert Users Simultaneously?

Set this parameter to True to alert every user in the workgroup at the same time. Set to False to alert users sequentially (one at a time) using the order configured for that workgroup in Interaction Administrator. See the Maintain Order option in the Interaction Administrator workgroup configuration container and the *CIC ACD Processing Technical Reference* located in the PureConnect Documentation Library for more information on configuring these linear hunt groups.

Linear hunt groups always alerts the first workgroup member as listed in Interaction Administrator. If that user does not pick up, or if the call could not be assigned to that user, then next user in the list is alerted. This means that users at the end of the list will receive calls less often.

Linear is not used if **Round Robin Alerting** is selected.

Round Robin Alerting?

Round robin alerting is similar to linear hunt groups, but CIC remembers which members have received calls and alerts the person who has not received a call for the longest amount of time. In Interaction Administrator, you can select the Maintain Order option for a list of workgroup members. If you select this option, the order of the members is used in deciding which user to alert. This option is ignored if **Alert Users Simultaneously** is selected.

For example, a workgroup has five users (User1 - User5), all available for workgroup calls. User1 is the first member of this workgroup as configured in Interaction Administrator. User2 is member 2, and so on.

If **Alert Workgroup** is invoked on call #1 and shortly thereafter on call #2, call #1 would alert on User1 and call #2 would alert on User2. If neither answers, call #1 will roll over to User3 and call #2 will roll over to User4. If User4 answers call #2 but User3 doesn't answer call #1, call #1 will roll over to User5. If call #1 is still not answered, it will roll over to User2. If there is still no answer, call #1 will roll over to User4 if User4 has finished with call #2, and will take the **Timeout** exit if not.

If at some later time another call invokes **Alert Workgroup**, the first user considered is not necessarily User1, but is the "next" user. That is, if User2 answered call #1 in the above scenario, the "next" user to try would be User3.

Alert Users if not logged in? option

Select this option to alert every member of the workgroup, even if they are not currently logged in. Stations are alerted for users not currently logged in. Clear this option to alert only members currently logged in.

Alert Users with other calls?

Set this option to True to alert users who are already connected to other calls. Connected user's see the call in their user queue, but their station will not alert because they are already connected to a call. Set to False to only alert users who are not connected.

Alert logged in Users with Do Not Disturb status?

Set this option to True to alert users who's status is set to a Do Not Disturb status. Set to False to only alert available agents.

Calling Number to Be Displayed?

Used to display information on ADSI phone or some other feature phone.

Calling Name to be displayed?

Used to display information on ADSI phone or some other feature phone.

Ring Cadence

Determines the ringing cadence used to alert the called party. The following table describes the values you can specify and the rings they produce:

Value	Ring produced: Ring On, (Ring Off)
1 (default)	2 seconds on, (4 seconds off)
2	1 seconds on, (5 seconds off)
3	.5 seconds on, (5.5 seconds off)
4	.5 seconds on, (.25 seconds off), .5 seconds on, (4.75 seconds off)
5	.5 seconds on, (.25 seconds off), .5 seconds on, (.25 seconds off), .5 seconds on, (4 seconds off)
6	.25 seconds on, (.25 seconds off), .25 seconds on, (.25 seconds off), .25 seconds on, (.25 seconds off), .25 seconds on, (4.25 seconds off)
7	1.25 seconds on, (.25 seconds off), .5 seconds on, (4 seconds off)
8	.5 seconds on, (.25 seconds off), 1.25 seconds on, (4 seconds off)

Note: When ringing a station with a phone type configured as Caller ID or ADSI (i.e., a phone that can display caller ID), TS will always use the default ring cadence (2 seconds on, 4 seconds off), regardless of which cadence is specified.

Frequency 1 in Hertz

The frequency of the tone you want to generate. Defaults to 350. To disable, set to 0.

Frequency 1 Amplitude in dB

The amplitude of the frequency you want to generate. Defaults to -10.

Frequency 2 in Hertz

The frequency of the tone you want to generate. Defaults to 440. To disable, set to 0.

Frequency 2 Amplitude in dB

The amplitude of the frequency you want to generate. Defaults to -10.

Call Waiting Tone Duration

The number of seconds the tone plays. To convert milliseconds to seconds, divide your millisecond value by 1000. For example, if you want the tone to play for 500 milliseconds, you would type .5 in this parameter.

Silence Duration After Call Waiting Tone

The number of seconds of silence before the call waiting tone repeats.

Call Waiting Tone Repeat Count

The number of times the call waiting tone plays before the call is transferred to voicemail.

Outputs

Keys

Contains any keys the caller pressed.

Exit Paths

Answered

This tool takes the answered exit path if the call was picked up by a member of the workgroup. The handler no longer has control of the call after that.

Escape

This tool takes the Escape exit path if the caller presses an escape key.

Timeout

This tool takes the Timeout exit path if the call alerts longer than the number of seconds specified in the Timeout parameter.

Failure

This tool can take the Failure exit path for several reasons. Failure can occur if the call disconnects, if the call is picked up by a user (and is no longer under the control of the handler), the call ID is no longer valid (if the call is deallocated), or system resource limitation.

Answer

This Telephony tool establishes a connection with an incoming call. This step fails if the Call Id is not valid.

Inputs

Call Identifier

The unique identifier for the incoming call.

Exit Paths

Next

After picking up a call, this step takes the Next exit path. Even if this step fails, it takes this exit path.

Assemble Prompt Phrase

This Telephony tool allows you to define a sequence of strings that defines the structure of a series of prompts you want to play. The specified prompts, TTS and .WAV files are combined into a string that can be played later with the [Play Prompt Phrase](#) tool.

Inputs

Call Identifier

The identifier for the call on which the prompt phrase is assembled. The tool uses the identifier to ascertain the language of the call.

Language

The language code of the prompt phrase to be played. If no language is specified, the default language will be used.

Singular if true

This Boolean permits simple handler-based control over the common case of generating different prompt sequences for plural arguments than for singular ones (e.g., "one car" vs. "two cars"). Set to True to select the Singular Prompt Sequence. Set to False to select the Plural Prompt Sequence.

Prompt Strings

This tab allows the specification of zero or more Prompt Strings that may need to be substituted dynamically into the phrase. Typically, the entries on this tab are string literals or the names of string variables. A Prompt String consists of one or more prompt identifiers separated by white space. A prompt identifier can be scoped by a handler name or unscoped. A scoped prompt identifier consists of the handler name, a colon and the prompt ID (e.g., <HandlerName:PromptName>). All identifiers are case insensitive.

Variables

List of String Variables in the current handler.

Parameters

The run-time substitution strings that are passed in at run-time to the sequence string.

String

The string variables must be either a </tts:....>, </file:.....> or a prompt in the form <PROMPT_NAME> for prompts contained within the handler, or <HANDLER:PROMPT_NAME> for prompts contained in another handler.

Sequence Strings

A sequence string is a mechanism by which you arrange multiple prompts to be played at one time.

In the simplest Case, the Sequence String would be <%1>. This means there will be one substitution string from the Prompt Strings dialog below. The Prompt Strings dialog will either pass in a prompt, or a string variable which represents a prompt. However, the Prompt Sequence could be much more complex than that, so here's a more detailed example:

```
<Prompt_IVR:IVR_REMOTE_VM_MENU> <%1> <IVR_REMOTE_VM_EXIT> </file=thankyou.wav> </tts=mister smith>
```

This will play the following items in the order shown:

1. Prompt IVR_REMOTE_VM_MENU from the Prompt_IVR handler
2. %1 - substitution string
3. Prompt IVR_REMOTE_VM_EXIT from the current handler
4. a .WAV file "thankyou.wav"
5. Text to Speech saying, "Mister Smith".

Single Sequence and Plural Sequence show what the sequence will be based on the Singular? parameter from the Inputs page.

This dialog lets you choose which sequence string to use for this phrase. The Handlers dialog lists the handlers you currently have open. Sequence Strings are resources just like prompts - they are resources of a handler. Simply highlight which handler you want to use and check the box of the sequence string you want to use.

The Delete, Edit, and Insert buttons allow you to manage existing sequence strings and create new ones. When creating new sequence strings, we highly recommend making use of the Notes field in the dialog box that appears as this will greatly facilitate any future editing.

String Output

Variable out

Exit Paths

Next

This tool always takes the Next exit path.

Assemble String from Attributes

This Telephony tool builds a string using a format string that can include attribute values from the given interface. This tool is used as part of the Interaction Attendant Set Attribute node.

Inputs

InteractionID

The interaction from which to draw any attribute values referenced in the given Format String.

Format String

A string that specifies how the result string should be build. This can include normal text and numbers, white space and punctuation characters. The special syntax for referring to an interaction attributes is \$(attribnam). For example, this format string: "http://myserver/popmyscreen/id?=\${myattribute}" where the interation's myattribute value is "9927" would yield this result after replacement: "http://myserver/popmyscreen/id?=9927"

Outputs

Result String

The string that contains all attribute references replaced by their current values. If an attribute does not exist, the attribute reference in the string contains blank.

Exit Paths

Success

This path is taken if the attributes were successfully replaced.

Failure

This path is taken if there was an error querying the interaction attributes.

See Also

For a list of interaction attributes, refer to the *Interaction Attributes Reference Guide* (attrib.chm) in the System APIs section of the PureConnect Documentation Library.

Assemble Text Phrase

This Telephony tool allows you to define a sequence of international strings. These strings are a text equivalent of voice prompts that can be use in chats, display strings, emails, faxes, etc.

Inputs

Language

The language code of the prompt phrase to be played. If no language is specified, the default language will be used.

Singular if true

This Boolean permits simple handler-based control over the common case of generating different prompt sequences for plural arguments than for singular ones (e.g., "one car" vs. "two cars"). Set to True to select the Singular Prompt Sequence. Set to False to select the Plural Prompt Sequence.

Text Strings

This tab allows the specification of zero or more international strings that may need to be substituted dynamically into the phrase. Typically, the entries on this tab are string literals or the names of string variables.

Variables

List of String Variables in the current handler

Parameters

The run-time substitution strings that are passed in at run-time to the sequence string.

String

These may be string literals or string variables.

Sequence Strings

A sequence string is a mechanism by which you arrange multiple prompts to be played at one time.

The dialog lets you choose which sequence string to use for this phrase. The Handlers dialog lists the handlers you currently have open. Sequence Strings are resources just like prompts - they are resources of a handler. Simply highlight which handler you want to use and check the box of the sequence string you want to use.

String Output

Variable out

Exit Paths

Next

This tool always takes the Next exit path.

Auto Conference

This Telephony tool creates a named conference using the specified call identifier. Further calls can be added to the same conference by calling this tool again with other call identifiers and specifying the same conference name. If a conference with the specified name does not exist, one is automatically created.

Inputs

Conference Name

A unique name for the conference – in this instance, the conference name for the conference to which you want to add the Call Identifier.

Note: A value is required for this input.

Conference PIN

The PIN or password the participant intends to use when entering the conference.

Call Identifier

The unique identifier (only one) to be added to the named conference.

Auto Disconnect

Set this option as follows:

- To automatically disconnect a call when it is the last one remaining in a conference, set this option to **Yes**.
- To maintain a connection for the last call in a conference, set this option to **No**.
- To allow the server parameter to determine which action to take, set this option to **Default**.

Exit Paths (Conference)

Success

This step takes the Success exit path if both the Call Identifier and the Conference Name are valid.

Failure

This tool can take the Failure exit path for several reasons. Failure can occur if the call disconnects, if the call is picked up by a user (and is no longer under the control of the handler), the call ID is no longer valid (if the call is deallocated), or there is a system resource limitation.

Bind Provisional Station

This Telephony tool binds a provisional station to a real station device. This is used in provisioning to assign the MAC address of the provisional station to the station chosen through the TUI.

Inputs

Provisional Station Queue Identifier

The unique identifier for the provisional station.

Real Station Queue Identifier

The unique identifier for the real station device.

Exit Paths

Success

This tool takes the Success exit path if the operation is successful.

Failure

This tool takes the Failure exit path if the operation cannot be performed.

Blind Transfer

This Telephony tool transfers a call (chat or telephone call) to an internal queue and generates a `CallOfferingNonSystemQueue` event. This event starts any handler with the Call to Non-System Queue initiator, such as `System_CallOfferingNonSystemQueue` handler. (This tool is different from the Transfer tool which transfers a call to another connected call.)

This tool will wait until the transferred call is connected. Once it is, any tracking that is being done on the call will terminate. If it is not answered, the call will be returned for additional processing.

Note: Although you can also use this tool to transfer a call to another number, its intended use is with internal queues. It is not used for external transfers in any of the handlers included with CIC and does not use the Dial Plan. For external transfers, see [Extended Blind Transfer](#).

Inputs

Call Identifier

The unique identifier for the call to be transferred.

Telephone Number (Queue identifier)

Enter any string expression, including a literal telephone number, a queue ID, or a complex expression built with the [Expression Editor Assistant](#). A comma causes a two-second pause, and any numbers after the "/" symbol are dialed after the call is connected.

You can transfer a call to both remote numbers and queues. If you transfer a call to a queue, you must use a scoped queue name such as "User Queue:MikeG".

Outputs

Outbound Call Identifier

The completed call identifier. If the call is transferred to an external number and the call cannot be completed via a put-back transfer (e.g., two B-channel transfer, Release Link Transfer, or flashhook transfer), then this output parameter will contain the callID for the external call when the transfer is completed. Otherwise, if the call transfers successfully, the value is always zero.

This parameter is always cleared, regardless of whether or not an inbound transfer is performed. Therefore this passed variable should not be reused later in the handler.

Exit Paths

Success

This step takes the Success exit path if the Call ID is valid.

Failure

This tool can take the Failure exit path for several reasons. Failure can occur if the call disconnects, if the call is picked up by a user (and is no longer under the control of the handler), the call ID is no longer valid (if the call is deallocated), or system resource limitation.

Alert Failure

This tool takes the Alert Failure path if there was a failure attempting to alert the object. This could be caused by the object not being of a type that the tool can alert or a failure in the alert itself.

Change Workgroup Queue Initiator

This Telephony tool allows users to change or indicate which outbound workgroup queue initiated a call. This tool was designed to be used with the Dial on Behalf of Workgroup feature.

Inputs

Call Identifier

The ID for the interaction for which the queue change is being made.

New Workgroup Queue Initiator

The new workgroup queue to use as the initiator of the call.

Exit Paths

Success

This step takes the Success exit path if the change is made successfully.

Failure

This tool takes the Failure exit path if TsServer was not able to perform the change because the call was disconnected, the call was deallocated, or TsServer failed to communicate with Queue Manager to perform the change.

Complete Parallel Make Call

This Telephony tool flags a successfully connected call to be placed on the List of Succeeded Calls generated with the [Parallel Make Call](#) tool.

Inputs

Call Identifier

The unique identifier for a call.

Exit Paths

Next

This tool always takes the Next exit path.

Compress Audio File

This Telephony tool uses EICAudioCompressor COM to create a compressed audio file from a specified source file.

Inputs

Source File Name

The name of the file to be compressed.

Destination File Name

The file you want to save the newly compressed file under.

Compression Format

An integer value indicating the type of compression to be applied to the file. Valid values are:

0 TrueSpeech

1 GSM

2 MS ADPCM - This format is unsupported so the tool will fail and an error will be logged.

3 G.711

4 G.726 - This format is unsupported so the tool will fail and an error will be logged.

5 IMA ADPCM - This format is unsupported so the tool will fail and an error will be logged.

See the *Voice Mail Compression Options* white paper for a more detailed explanation of audio file compression and compression formats.

Normalize

Set this Boolean to True to normalize the audio file. When a recording is normalized, it is analyzed to determine what the maximum volume level of the audio file is. A value 5% below the maximum value is then used to set the gain value that will bring the maximum volume level up or down to a standard level. This ensures that at a given station, all recordings will play back at the same relative volume. Using a value 5% below the maximum volume to calculate the gain prevents a short burst of static or similar anomalous noise from throwing off this volume adjustment.

Delete Source

The source file will be deleted once the new file is successfully saved if this box is checked. This box is not checked by default.

Overwrite Destination

The new file will automatically overwrite a pre-existing file with the same name as the destination filename if this box is checked. This box is checked by default.

Timeout

The number of seconds the tool should wait for the file to be compressed. If no value is returned in the specified time, this tool will take the Failure exit path.

Exit Paths

Success

This path is taken if the new audio file is successfully processed.

Failure

This path is taken if the operation fails.

Conference

This Telephony tool step creates a conference call by connecting together two calls. (Add more calls (beyond the first two) with the [Add Party](#) tool.)

Inputs

Call Identifier

The unique identifier for a call. In this instance, a call to a party you want to conference.

Call Identifier

The unique identifier for a second call. In this instance, a call to a second party you want to conference.

Auto Disconnect Last Call

Enabling this feature will automatically disconnect a call if it is the last one remaining in a conference. Auto Disconnect Last Call may be set to "Yes" if this is desired, "No" if you want the last call in a conference to remain connected, or "Default" to allow the server parameter to determine which action to take.

Maximum Number of Parties

The maximum number of parties allowed in the conference. Default is -1. This information can be used for load balancing or server selection.

Outputs

Conference Call Identifier

The unique identifier for the conference call created from the two objects on the Inputs page.

Exit Paths (Conference)

Success

This step takes the Success exit path if both the Call Identifier and the Conference Identifier are valid.

Failure

This tool can take the Failure exit path for several reasons. Failure can occur if the call disconnects, if the call is picked up by a user (and is no longer under the control of the handler), the call ID or conference ID is no longer valid (if the call is deallocated), or system resource limitation.

Convert Call Id to String

This Telephony tool converts an interaction's Call ID into a string.

Inputs

Interaction Id

The unique identifier for an interaction.

Outputs

String

The string representing the interaction's Call ID.

Exit Paths

Next

This tool always takes the Next exit path.

Convert Conference ID to String

This Telephony tool converts an interaction's conference ID into a string.

Note: This tool replaces the Conference ID to Integer tool. Custom handlers containing the Conference ID to Integer tool will continue to work but new handlers must use the Convert Conference ID to String tool.

Inputs

Conference Id

The unique identifier for a conference interaction.

Outputs

String

The string representing the interaction's conference ID, which is a 32-bit number.

Exit Paths

Next

This tool always takes the Next exit path.

Convert String to Call Id

This Telephony tool is the reciprocal of [Convert Call Id To String](#), converting a string variable back into a valid integer Call ID.

Inputs

Interaction Id String

The string representing the interaction's Call ID.

Outputs

Interaction Id

The unique identifier for the interaction.

Exit Paths

Success

This path is taken if the string is successfully converted.

Invalid

This path is taken if the specified Call ID string is invalid.

Convert String to Conference ID

This Telephony tool is the reciprocal of [Convert Conference Id To String](#), converting a string variable back into a valid integer Conference ID.

Note: This tool replaces the Integer to Conference ID tool. Custom handlers containing the Integer to Conference ID tool will continue to work but new handlers must use the Convert String to Conference ID tool.

Inputs

Conference Id String

The string representing the interaction's Conference ID.

Outputs

Conference Id

The unique identifier for a conference interaction.

Exit Paths

Success

This path is taken if the string is successfully converted.

Invalid

This path is taken if the specified Conference ID string is invalid.

Deferred Answer

This Telephony tool causes CIC to store the information needed to establish a connection with a specified incoming call at a later time but does not establish the connection immediately as the [Answer](#) tool step does.

This tool enables Interaction Processor to delay answering an incoming call until a valid agent has been reached or the IVR is entered so that incoming callers have time to disconnect without being charged by their telecommunications provider.

Inputs

Call Identifier

The unique identifier for the incoming call.

Implicitly answer call when required?

If True, causes CIC to automatically establish a connection with the specified call when most media requests occur. For example, if this option is enabled and Interaction Processor (handlers) sends a call to the IVR which requires the call to be answered to play an audio file, the call will be answered before the play occurs. Note that ringback audio triggered by the alert tool step does not trigger this option because this would result in the call being answered almost immediately in common scenarios, defeating the purpose of the feature.

If False, the tool step acts primarily as a placeholder to indicate intention on the part of the person who wrote the handler routine.

Exit Paths

Next

This tool step always takes the Next exit path.

Describe Phone Number

This Telephony tool processes a telephone number to determine where it falls within your dial plan. See the *Dial Plan* technical reference document in the PureConnect Documentation Library for more information on configuring dial plans. Also see the related *CIC Regionalized Dial Plan* technical reference document for information about regionalized dial plans using locations.

Inputs

Input Phone Number

The telephone number to process.

Location Filter

Specifies a string that indicates a Location Filter specified in the Dial Plan. By default, the value should be "" (an empty string), but there may be many custom Location Filters listed as well, if they were previously defined in the Regionalization configuration. Enter the Location Filter name as it appears in the dial plan in Interaction Administrator.

Outputs

Dial Groups

A dial group is a group of lines or channels that CIC uses for outbound calls.

Dial Strings

The dial string is the actual string of characters sent to the CO to place a call.

Classifications

The classification of the input phone number. The classification name might represent all calls of a certain format or location, such as "Local", "Long Distance", "Emergency", etc. Classifications are configured in Interaction Administrator. For more information on classifications, see the Interaction Administrator online help.

Classification Categories

The classification category of the phone number. A classification category is a non-translatable string that represents one or more phone number classifications. Classification categories are configured in Interaction Administrator. For more information on classification categories, see the Interaction Administrator online help.

Pattern Name

The name associated with the pattern.

Display String

The display string is the formatted phone number that is displayed on a CIC client interface.

Default Dialstring

The dialstring used if a classified dialstring cannot be found.

Default Classification

The classification used if a classification is not specified or cannot be found.

Default Classification Category

The classification category used if one is not specified or cannot be found. See *Classification Categories* above.

Country Code

The country code associated with the number.

Standardized Number

A consistent format for the type of number. A more in-depth discussion of standardized numbers can be found the *Dial Plan* technical reference document in the PureConnect Documentation Library.

Account Code Required

A boolean indicating whether or not a required account code is associated with this number.

Logical Components

Allows you to specify user-defined segments/components. A more in-depth discussion of standardized numbers can be found the *Dial Plan* technical reference document in the PureConnect Documentation Library.

Logical Component Names

Allows you to specify a label for the user-defined segments/components. A more in-depth discussion of standardized numbers can be found the *Dial Plan* technical reference document in the PureConnect Documentation Library.

User Data

Any data to add to the User Data field. This can be any string.

Exit Paths

Next

This tool always takes the Next exit path.

DID/DNIS Routing Ex

This Telephony tool looks up the queue name to which you can route a telephone number.

Inputs

Telephone Number

A string that contains the telephone number for which to look up the queue.

Location (for future use)

The tool does not use this input.

Search DID/DNIS Routing Table

Set this Boolean parameter to True to search the DID/DNIS Routing Table.

Search Queue Extensions

Set this Boolean parameter to True to search the Queue Extensions list.

Search Queue Name

Set this Boolean parameter to True to search the Queue Names list.

Outputs

Scoped Queue Name

The queue name for the telephone number.

Exit Paths

Success

This path is taken when the queue name is successfully retrieved.

Failure

This path is taken when the queue name is not retrieved.

Disconnect

This Telephony tool disconnects a telephone call.

Inputs

Call Identifier

The unique identifier for the telephone call to be disconnected.

Cancel Pending Operations?

This Boolean determines whether or not any pending operations will be cancelled when the call is disconnected.

Reason Code

An integer representing the reason that the call was disconnected. The following reason codes are currently used by this tool:

0	None
1	No answer to the call.
2	The line was busy.

Exit Paths

Next

This step always takes the Next exit path.

Extended Blind Transfer

This Telephony tool transfers a call to a number. It also monitors the call to see if it gets through.

This tool will wait until the transferred call is connected. Once the call is connected, any tracking that is being done on the call will terminate. If it is not answered, the call will be returned for additional processing.

Note: It is expected that the dial plan has already been executed prior to this tool step.

Inputs

Call Identifier

The unique identifier for a call.

Telephone Number

The queue id or phone number of the transfer recipient. A comma causes a two-second pause, and any numbers after the "/" symbol are dialed after the call is connected.

You can transfer a call to both remote numbers and queues. If you transfer a call to a queue, you can use both a scoped or non-scoped queue name. Non-scope Queue name refers to the name of the queue object without the queue type; for example, a phone is referred to as "User Queue:MikeG" (a scoped queue name) and the Non-Scoped name is just "MikeG". If you use a non-scoped queue name, CIC assumes the queue is a User queue.

Lines Groups (empty list means any line)

This parameter takes a list of line groups (as configured in Interaction Administrator), or leave this parameter empty to use any available line group.

List of dial strings to be used

This list of dial strings parallels the list of line groups from the parameter above. When line group from position one is attempted, dial string from position one is used, and so on. In the default shipping handlers, the DialPlanEX subroutine returns a list of dial strings that parallels the list of line groups. You can pass the list of dial strings from that subroutine into this parameter. A comma in a dial string causes a two-second pause, and any numbers after a "/" symbol are dialed after the call is connected.

Calling Party Number

This parameter passes a string of digits to be displayed as ANI or Caller ID on the phone system that receives the call. If you do not specify a value in this parameter, the Telephony Services subsystem uses the number entered in the Phone Number field in the Line Configuration container in Interaction Administrator.

Note: This only works for ISDN.

Calling Party Name

This parameter passes a string to display the name of the caller on the phone system that receives the call. If you do not specify a value in this parameter, the Telephony Services subsystem uses the name associated with the number entered in the Phone Number field in the Line Configuration container in Interaction Administrator.

Note: This only works for ISDN.

Use Putback (if available)

This Boolean is used to determine whether or not to use the putback feature if it is available for that line.

Diversion Number

Used for forwarded SIP calls, this field specifies the destination/address to which the call was originally sent.

Diversion Name

A name for the SIP diversion address.

Diversion Reason

A number representing one of the SIP diversion reasons:

Value	Description
0	None
1	Unknown
2	Busy
3	No answer
4	Unavailable
5	Unconditional
6	Time of day
7	Do not disturb
8	Deflection
9	Follow-me
10	Out of service
11	Away
255	Other

Call Attribute Names

The names of any attributes to be assigned to the call as Queue Manager attributes before the Make Call is placed on TS.

Call Attribute Values

The values of any attributes named in the Call Attribute Names parameter above.

Outputs

Outbound Call Identifier

If the call is transferred to an external number and the call cannot be completed via a put-back transfer (for example, two B-channel transfer, Release Link Transfer, or flashhook transfer), then this output parameter will contain the call ID for the external call when the transfer is completed. Otherwise, if the call transfers successfully, the value is always zero.

This parameter is always cleared, regardless of whether or not an inbound transfer is performed. Therefore, this variable should not be reused later in the handler.

Exit Paths

Success

This tool takes the Success exit path if no other exit is taken.

Failure

This tool can take the Failure exit path for several reasons. Failure can occur if the call disconnects, if the call is picked up by a user (and is no longer under the control of the handler), the call ID is no longer valid (if the call is deallocated), or system resource limitation.

Alert Failure

This tool takes the Alert Failure path if there was a failure attempting to alert the object. This could be caused by the object not being of a type that the tool can alert or a failure in the alert itself.

Extended Get Key

This Telephony tool step accepts telephone keypad presses from a caller and takes an exit path based on those keys. This step also listens for a fax tone. This tool is often used in IVR handlers and subroutines to gather a caller's response to a prompt.

Use the [Get Key](#) tool if you just want to collect one key press and don't want to detect tones.

Inputs

Call Identifier

The unique identifier for a call.

Valid Keys

0,1,2,3,4,5,6,7,8,9 are the keys a caller may enter. You can change these keys using the [Expression Editor Assistant](#).

Escape Keys

"*" is the escape key. When a caller presses the escape key, this step stops accepting caller input and takes the Escape exit path. Change these keys using the Expression Editor Assistant.

Termination Keys

"#" is the termination key for this step. This step immediately stops waiting for key input when this key is pressed and exits through the Success exit path. If the caller presses the termination key before entering any other keys, this tool takes the Success exit path. The termination key appears in the Digits output key if it was included in the Valid Keys input. Otherwise, the Digits output variable is empty.

Maximum number of Keys

The maximum number of keys this step accepts before taking the Successful exit path.

Maximum Inter-Digit Delay

The number of seconds to allow between key presses before this step takes the Success exit path. The default is 2.5 seconds. The Inter-Digit delay is not counted until the caller presses the first key. If the inter-digit delay expires, the tool will exit via the Success exit path with an output of whatever digits the caller pressed up to that point.

Note for the following Tone Detection parameters: The default values for the next 8 settings recognize a standard fax tone. In almost all cases, you do not need to change the default settings. If you want to detect a tone other than a fax device, refer to the documentation for the device generating the tone. Also see the topic on [Tones](#).

Tone Detection Frequency 1 in Hertz

1100 To disable Fax tone detection, set this to 0.

Permissible deviation in Tone Detection Frequency 1

50

Tone Detection Frequency 2 in Hertz

0

Permissible deviation in Tone Detection Frequency 2

50

Timeout

The total number of seconds after audio is played for any keys to be entered before this tool takes the Timeout exit path. The default for this parameter is 20 seconds. If you are using this Extended Get Key step simply to flush any queued audio, set this value to 0 seconds. You should not set this value to zero unless you are flushing audio. The maximum value for this tool is 600 seconds. Any value specified greater than 600 will cause this tool to wait for 600 seconds.

This tool accepts a caller's key presses while the audio is being played.

Attribute to monitor as a termination event

This optional parameter specifies the name of an attribute to be monitored during the operation of this tool. If the value of this attribute changes during the Extended Get Key operation, the tool will exit taking the Attribute exit path.

Suppress logging of the input digits

Set this to True if you do not want the input digits to be recorded in the trace file. This should be done if the digits being entered are security-related digits, such as PINs or passwords.

Interaction ID for the attribute monitor

This optional parameter specified the interaction ID associated with the attribute to monitor.

Repeat queued plays if audio interruption is encountered?

If set to true, queued plays are repeated if audio is interrupted (a voice mail greeting is detected).

Outputs

Keys

This is the variable that holds the caller's key input. This variable is empty if the caller pressed the termination key without entering any digits.

Exit Paths

Success

This step takes the Success exit path if the digits are received. If the caller pressed the termination key (#) without entering any digits, this tool takes the Success exit path and the Digits output variable is empty.

Escape

This step takes the Escape exit path if the caller presses the configured escape key.

Tone

This step takes the Tone exit path if it detects the tone specified by the Tone parameters on the Inputs pages.

Timeout

This step takes the Timeout exit path if no keys are pressed within the amount of time allotted in the Timeout parameter on the Inputs page. The GetQueueAnnouncement handler, which uses this tool step and calls the GetDigit function in Telephony Services,

also takes this exit path if it encounters an error and terminates with an unknown reason.

Failure

This tool can take the Failure exit path for several reasons. Failure can occur if the call disconnects, if the call is picked up by a user (and is no longer under the control of the handler), the call ID is no longer valid (if the call is deallocated), or system resource limitation.

Attribute

This path is taken if an attribute that has been set to be monitored changes during the operation of this tool.

Extended Get Key Async

This Telephony tool works much like the [Extended Get Key](#) tool, except that it will not pause the handler to wait for the digits to be received. This tool will also recognize a fax tone. Once this tool exits, no further processing will be done on the call by the handler. When the requested keys are received, the [Get Digits Ex Async](#) initiator will launch a separate handler to continue call processing.

Inputs

Call Identifier

The unique identifier for the call.

Valid Keys

0,1,2,3,4,5,6,7,8,9 are the keys a caller may enter. You can change these keys using the [Expression Editor Assistant](#).

Escape Keys

"*" is the escape key. When a caller presses the escape key, this step stops accepting caller input and takes the Escape exit path. Change these keys using the Expression Editor Assistant.

Termination Keys

"#" is the termination key for this step. This step immediately stops waiting for key input when this key is pressed and exits through the Success exit path. If the caller presses the termination key before entering any other keys, this tool takes the Success exit path and the Digits output variable is empty.

Maximum Number of Keys

The maximum number of keys this step accepts before taking the Successful exit path.

Maximum Inter-Digit Delay (seconds)

The number of seconds to allow between key presses before this step takes the Success exit path. The default is 2.5 seconds. The Inter-Digit delay is not counted until the caller presses the first key. If the inter-digit delay expires, the Get Digits Ex Async initiator will then launch with a Termination Code of Success.

Note for the following Tone Detection parameters: The default values for the next 8 settings recognize a standard fax tone. In almost all cases, you do not need to change the default settings. If you want to detect a tone other than a fax device, refer to the documentation for the device generating the tone. Also see the topic on [Tones](#).

Tone Detection Frequency 1 in Hertz (0:disable)

1100 To disable Fax tone detection, set this to 0.

Permissible deviation in Tone Detection Frequency 1 (Hz)

50

Tone Detection Frequency 2 in Hertz (0:disable)

0

Permissible deviation in Tone Detection Frequency 2 (Hz)

50

Timeout (seconds)

The total number of seconds to wait for any input after audio is played before the get key operation terminates. The Get Digits Ex Async initiator will then launch with a Termination Code of Timeout. The default for this parameter is 20 seconds.

Attribute to monitor as a termination event

This optional parameter specifies the name of an attribute to be monitored during the operation of this tool. If the value of this attribute changes during the Extended Get Key operation, the Get Digits Ex Async initiator will launch with a Termination Code of Attribute Changed.

Correlation ID

Specifies which handler this tool is in. The Get Digits Ex Async initiator will use this information to resume processing once get digits operation ends.

No Response Timeout (seconds)

The number of seconds to wait for any response before the get key operation terminates. The Get Digits Ex Async initiator will then launch with a Termination Code of No Response. The default for this parameter is 20 seconds.

Interaction ID for the attribute monitor

This optional parameter specifies the interaction ID associated with the attribute to monitor.

Exit Paths

Next

This tool always takes the Next exit path.

Extended Place Call

This Telephony tool step places a call on an outside line. Extended Place Call has many exit paths to handle the different possible outcomes of this step.

Inputs

Telephone Number

If the **List of dial strings to be used** parameter (described below) is used, this parameter indicates the telephone number to display to a CIC Client user. This number is not dialed.

If the **List of dial strings to be used** parameter is empty, then the telephone number in this parameter is used to dial the call.

Lines Groups

This parameter takes a list of line groups (as configured in Interaction Administrator), or leave this parameter empty to use any available line group. In the default shipping handlers, the DialPlanEX subroutines returns a list of line groups to be used. You can pass that list in this parameter.

Station Queue Identifier

If this string is empty, the call is placed from the system queue. If the string identifies a station ("Station:Gagle", "Station Queue:Gagle" or just "Gagle"), the call is placed from the named station.

Workgroup Queue Identifier

Identifies the workgroup from which this call originated. This parameter is useful for auto-dialers, such as the Interaction Dialer, when the ACD Server determines which agent a connected call should be sent to.

List of dial strings to be used

This list of dial strings parallels the list of line groups from the parameter above. When line group from position one is attempted, dial string from position one is used, and so on. In the default shipping handlers, the DialPlanEX subroutine returns a list of dial strings that parallels the list of line groups. You can pass the list of dial strings from that subroutine into this parameter. A comma in a dial string causes a two-second pause, and any numbers after a "/" symbol are dialed after the call is connected.

Perform Call Analysis

Set this Boolean parameter to "True" if you want the handler to detect busy signals or whether the party did not pick up. Set to "False" if you are integrating with a PBX that does not generate the expected call progress audio. See [Call Analysis](#) for more information.

Note: In-band voice messages (such as "This is AT&T, it is 4 P.M. where you are calling") can cause call analysis to fail. If this occurs, contact your long distance provider to have this message removed.

Note: If you turn off call analysis, calls made over analog, T1, and E1 lines will take the success exit path even if the call is busy. See [Call Analysis](#) for more information.

Include Answering Machine Detection in Call Analysis?

Set this to True to have CIC attempt to detect an answering machine. If a machine is detected, this tool takes the Machine exit path. If this is set to False, answering machines are not detected. See [call analysis](#) for more information on answering machine detection.

Silence Wait Timeout

The number of seconds of silence to elapse before taking the Machine exit path. This is useful if you want to ensure that the machine's greeting has finished playing before you play an audio file to the machine. This field is ignored if the **Include Answering Machine Detection** option is cleared.

No Answer Timeout

The number of seconds to wait before taking the No Answer exit path. The default value is 30 seconds. Fractions of seconds are not valid. The timeout value must be 1 or greater. Setting the No Answer Timeout value to 0 actually causes it to default to 30 seconds.

Note: To use the **No Answer Timeout** parameter, you must also select the **Honor No Answer Timeout** parameter in Interaction Administrator. If the **Honor No Answer Timeout** parameter is not selected, the default value of 30 seconds is used for the **No Answer Timeout** parameter. For more information about the **Honor No Answer Timeout** parameter, see [General Telephony Parameters](#) in Interaction Administrator help.

Calling Party Number

This parameter passes a string of digits to be displayed as ANI or Caller ID on the phone system that receives the call. If you do not specify a value in this parameter, the Telephony Services subsystem uses the number entered in the Phone Number field in the Line Configuration container in Interaction Administrator.

Note: This only works for ISDN.

Calling Party Name

This parameter passes a string to display the name of the caller on the phone system that receives the call. If you do not specify a value in this parameter, the Telephony Services subsystem uses the name associated with the number entered in the Phone Number field in the Line Configuration container in Interaction Administrator.

Note: This only works for ISDN.

Use the dial plan?

Set to True if the call is an intercom call.

Note: If set to True, this tool will not retrieve voice mail settings. This functionality will be added in a future release.

Create the call with the CallInitiationContext attribute set to this value

This parameter sets the CallInitiationContext attribute to the specified value.

Call for which this is a consulting call (optional)

If this is a consulting call, the call ID for the call being consulted may be placed here.

Call Attribute Names

The names of any attribute to be assigned to the call as Queue Manager attributes before the Make Call is placed on TS.

Call Attribute Values

The values of any attributes named in the Call Attribute Names parameter above.

Diversion Number

Used for forwarded SIP calls, this field specifies the destination/address to which the call was originally sent.

Diversion Name

A name for the SIP diversion address.

Diversion Reason

A number representing one of the SIP diversion reasons:

Value	Description
0	None
1	Unknown
2	Busy
3	No answer
4	Unavailable
5	Unconditional
6	Time of day
7	Do not disturb
8	Deflection
9	Follow-me
10	Out of service
11	Away
255	Other

Outputs

Call Identifier

The unique identifier for the outbound call.

Exit Paths

Success

This step takes the Success exit path if the call goes through successfully. If call analysis is turned off, all successfully dialed calls are considered successful if the call connects, even if the connection is a busy signal. (Note: Calls made over ISDN lines may still take the Busy exit path if a D-channel message is returned indicating that the dialed number is busy.)

Busy

This step takes the Busy exit path if the called number is busy, or if there are no available outgoing lines.

No Answer

This step takes the No Answer exit path if the called number does not answer.

Not Reached

This tool takes the Not Reached exit path if there is no dial tone or ring back.

Intercept

This tool takes the Intercept exit path if special information tones are detected. This can occur when an invalid number is reached, and the external telephone system plays a recorded message.

Machine

This tool takes the Machine exit path if call analysis is turned on and an answering machine is detected.

Fax

This tool takes the Fax exit path if call analysis is turned on and a fax tone is detected.

No Lines

This tool takes the No Lines exit path if no lines are available to place this call.

Disconnect

This tool takes the Disconnect exit path if the call is disconnected prior to completion.

Failure

This tool can take the Failure exit path for several reasons. Failure can occur if the call disconnects, if the call is picked up by a user (and is no longer under the control of the handler), the call ID is no longer valid (if the call is deallocated), or system resource limitation.

Canceled

This tool takes the Canceled exit path if the call attempt is cancelled prior to completion.

Flush Audio

This Telephony tool forces any prompt that has been queued to play immediately.

Inputs

Call Identifier

The unique identifier for the call playing the audio file.

Wait for Play Completion

This field determines whether or not the caller may interrupt this prompt by pressing keys on the keypad. Setting it to False means that this prompt may be interrupted. Setting this parameter to True will cause the prompt to play and not be interrupted.

Exit Paths

Next

This tool always takes the Next exit path.

Flush Keys

This Telephony tool clears from memory any keys a caller has entered up to this point.

Inputs

Call Identifier

The call to have key presses cleared from memory.

Exit Path

Next

This step always takes the Next exit path.

Get Attribute

This Telephony tool step obtains an attribute from an interaction and assigns that attribute as the value of a variable. If that call object attribute does not exist, the value in the output variable is null. For a list of the call object attributes that can be retrieved, see the *Interaction Attributes Technical Reference* document in the PureConnect Documentation Library.

Note: This tool cannot get a value for an attribute in a Directory Services key. To get a value in a Directory Services Key, use the [GetDsAttr](#) or [GetDsAttrs](#) tools.

Inputs

Call Identifier

The ID for the interaction to query.

Call Attribute Name

The name of the attribute this step retrieves.

Outputs

Call Attribute Value

The variable that contains the value of the retrieved attribute.

Exit Paths

Next

This step always follows the Next exit path.

Get Attributes

This Telephony tool retrieves a list of attributes for an interaction and the value of the attributes. These can be default call attributes or custom attributes you created with the [Set Attribute](#) tool. You specify which attributes to return, and the values are output in a parallel list of strings. For example, if CallID is element 0, the returned value will also be in element 0.

Inputs

Call Identifier

The ID for the interaction to query.

Call Attribute Names

The attributes to retrieve. For a list of attributes that can be retrieved with this tool, see the *Interaction Attributes Technical Reference* in the PureConnect Documentation Library.

Outputs

Call Attribute Values

The parallel list of values for the attributes you specified.

Exit Paths

Next

This step always takes the Next exit path.

Get Billing Rate

This Telephony tool retrieves the billing rate information stored for a call.

Inputs

Call Identifier

The identifier for the call that contains billing rate information. This billing rate information was created with a [Set Billing Rate](#) tool.

Outputs

Billing Rate String

This variable contains the billing information sent to AT&T's Vari-A-Bill services.

Exit Paths

Success

This step takes the Success exit path if the information was retrieved.

Failure

This tool can take the Failure exit path for several reasons. Failure can occur if the call disconnects, if the call is picked up by a user (and is no longer under the control of the handler), the call ID is no longer valid (if the call is deallocated), or system resource limitation.

Get Call Log

This Telephony tool obtains the contents of a Call Log attribute and writes it to the variable you specify in the Log Contents parameter. The information in the Call Log was inserted with the [Log Message](#) tool. This tool works for chat and telephone calls.

Inputs

Call Identifier

The unique identifier for the call.

Call Log Identifier

The name of the log this step is to retrieve.

Outputs

Call Log Contents

The variable that contains the value of the log file contents.

Exit Paths

Next

This step always takes the Next exit path.

Get Datetime Attribute

This Telephony tool retrieves the datetime value of a specified attribute for a call.

Inputs

Call Identifier

The unique identifier for the call.

Call Attribute Name

The attribute for which the datetime value is being retrieved.

Outputs

Call Attribute Datetime Value

The datetime value for the specified attribute.

Exit Paths

Next

This tool always takes the Next exit path.

Get Datetime Attributes

This Telephony tool retrieves the datetime value of multiple attributes for a specified call.

Inputs

Call Identifier

The unique identifier for the call.

Call Attribute Names

The list of attributes for which the datetime values are being retrieved.

Outputs

Call Attribute Datetime Values

The list of datetime values for the specified attributes.

Exit Paths

Next

This tool always takes the Next exit path.

Get Key

This Telephony tool step accepts a single key from a telephone keypad as input from a user and assigns that key as the value of the output variable. If you need to collect more than one key from a caller for a given operation, such as for a last name lookup, use [Extended Get Key](#). If the user presses an escape key, the escape key pressed is assigned as the value of the output variable.

The value written to the output variable (the keys a caller presses) replaces any previously existing value.

Inputs

Call Identifier

The unique identifier for a call the key input comes from.

Valid Keys

Keys that are allowed as input. You can change the valid keys using the [Expression Editor Assistant](#).

Escape Keys

The key a user can press to exit this step along the Escape exit path. You can leave this parameter empty, or have multiple escape keys. This step immediately stops waiting for key input when an escape key is pressed. You can change the escape key(s) using the [Expression Editor Assistant](#).

Timeout

The maximum number of seconds this step waits before taking the Escape exit. The value of this parameter is set to 600 seconds (10 minutes) by default. Setting the Timeout to 0 or a negative number causes this tool to wait indefinitely. Specifying any value greater than 600 results in the default of 600 seconds.

Suppress logging of the input digits

Set this to True if you do not want the input digits to be recorded in the trace file. This should be done if the digits being entered are security-related digits, such as PINs or passwords.

Outputs

Keys

The variable that holds the caller's key input.

Exit Paths

Success

If this step executes successfully, it exits through the Success exit path and the key pressed by the user is assigned to the output variable.

Escape

If the user enters an Escape key, it exits through the Escape exit path and the escape key pressed by the user is assigned to the output variable. This tool also takes the Escape exit path if the Timeout value expires.

Failure

This tool can take the Failure exit path for several reasons. Failure can occur if the call disconnects, if the call is picked up by a user (and is no longer under the control of the handler), the call ID is no longer valid (if the call is deallocated), or system resource limitation.

Get Station Info

This Telephony tool returns True or False to indicate if the SIP (i.e., Polycom) phone station queue has shared line appearances.

Input

Station Queue Identifier

The station queue name. For example: "Station Queue: KevinkSIP" This applies only to Polycom phone stations.

Outputs

Does station support Shared Line Appearances?

The default is "HasSharedAppearances," which is a boolean variable.

It is set to True if the station has shared appearances and false otherwise.

Exit paths

Success

If it succeeds in determining if the station does or does not have shared line appearances.

Failure

If it can not find the station queue identifier by the given name.

If the system detects an Invalid API Version (e.g., if the tool DLL is not the right version for the system)

If there is a timeout communicating with Telephony Services via Notifier (e.g, the system is bogged down)

If the station is Not Active

Get User's Location

This Telephony tool returns the location of the specified user, based on the location of the user's logged in station. If the user is not logged in or the user is logged into a remote station, the user's configured location is used. If the user does not have a configured location, the CIC server location is used.

Inputs

User

The user for which you want to determine the location.

Outputs

Location

The returned location.

Exit Paths

Success

This path is taken if the location was found.

Failure

This path is taken if the location was not found.

Get Users with Role

This Telephony tool returns a list of users with the specified role.

Inputs

Role

The role to search.

Include Inherited

If True, includes roles inherited from groups, etc. (defaults to True).

Qualified User IDs

If true, returns a list of fully scoped user IDs and the list is prepended with "User Queue:". If false, returns a list of user names.

Outputs

List of Users

User IDs (list of string).

Exit Paths

Success

This path is taken if the role was found and the list was returned, even if no users have that role.

No Results

This path is taken if the role was not found.

Failure

This path is taken if there is an internal error, such as the server not running.

Get Wildcard Attributes

This Telephony tool allows you to specify a wildcard value. It returns a list of call attribute names that matched the wildcard, and a corresponding list of the attribute values. This tool works for telephone calls and chat sessions.

Note: Only user-defined attributes are supported, not the default attributes created by Telephony Services, and wildcard matching is limited to the beginning of the attribute names. For example, if you use the wildcard value "IDS_", the following attributes have been defined "IDS_ONE," "IDS_TWO," "MY_IDS_FOO;" then the tool will return IDS_ONE and IDS_TWO. For more information, refer to the *Interaction Attributes Technical Reference* in the PureConnect Documentation Library.

Inputs

Call Identifier

The identifier for the call that contains the attributes you want to query.

Call Attribute Name Wildcard

The string value to search for. Wildcards can only represent the beginnings of attribute names. See the example in the note above for more info.

Outputs

Call Attribute Names

A list of names of the call attribute names.

Call Attribute Values

A parallel list of values for the attribute names.

Exit Paths

Next

This tool always takes the Next exit path.

Hold

This Telephony tool step places a call on hold. The call that is placed on hold must be in a Connected state.

Inputs

Call Identifier

The unique identifier for a call to be placed on hold.

True: Claim control of this call

Set to True to place the call on hold and pass control of the call back to a handler, even if it was already picked up by a user.

Set to False if you want this step to fail if a user has already picked up the call. The user maintains control of the call.

Exit Paths

Success

This step takes the Success exit path if the call is placed on hold.

Failure

This tool can take the Failure exit path for several reasons. Failure can occur if the call disconnects, if the call is picked up by a user (and is no longer under the control of the handler), the call ID is no longer valid (if the call is deallocated), or system resource limitation.

Is Alertable?

The purpose of this Telephony tool is to determine whether a call on a workgroup, user, or station queue may alert that queue. Once this tool has determined whether the call may alert on the queue, use the exit paths to process the call.

Note: This tool should not be used if the call is, or is going to be, flagged for ACD processing.

Workgroup Queues

A call can alert on the workgroup queue (and take the Success exit path from this tool) if:

- at least one workgroup member is logged in and monitoring the specified workgroup queue.
- at least one workgroup member has an available status.
- at least one workgroup member is not off hook.

User Queues

A call can alert on a user queue (and take the Success exit path from this tool) if:

- that user is logged in and monitoring that queue.
- that user has an available status.
- that user is not off hook.

Station Queues

A call can alert on a station queue (and take the Success exit path from this tool) if:

- that station does not have an active call or a call on hold.
- that station is not offhook.

Inputs

Queue Identifier

The identifier for the queue to query.

Examples:

"Workgroup Queue:Support"

"User Queue:JohnD"

"Station Queue:JohnDPC"

Outputs**Status Message Name**

Contains a valid result when the input (Queue Identifier) is a user queue. When valid, it contains the user's current status, such as Available, Gone Home, or Out of Office.

Date and Time for status

Valid only when querying a user queue set to a status with a return time, such as "On Vacation" or "Out of the Office." This is the date and time the user has designated as the return time. The value of this variable can be played as a prompt for the caller as part of an IVR handler. This value is invalid for workgroup and station queues.

Is date in StatusDateTime set?

Valid only when querying a user queue, this Boolean variable has a value of true if the date for the status is set. This value is invalid for workgroup and station queues.

Is time in StatusDateTime set?

Valid only when querying a user queue, this Boolean variable has a value true if the time for the status is set.

List of stations which should be alerted

This list of string value contains a list of all of the stations to be alerted. This station list should be explicitly checked for at least one value before retrieving the first element. It is possible for Is Alertable? to return Success even when this list is empty. This can occur if the user's Ring Phone attribute is false or if a workstation's Ring Always attribute is false.

Workgroup Queues

Is Alertable? creates a list of all stations associated with available workgroup members who are not off hook and in an available status.

User Queues

Is Alertable? creates a list of all station info into which the user is logged. This includes workstations connected to a computer on which the user has logged into a CIC client, stations through which a user has logged in remotely. This value is filled only if that user queue is alertable.

Station Queues

Is Alertable? adds the identifier for the queue being queried if that station queue is alertable.

Monitoring Stations

The list of stations that are monitoring the queue specified in the Queue Identifier input parameter.

Monitoring Users

The list of users who are monitoring the queue specified in the Queue Identifier input parameter.

Remote Telephone Number

Returned only when querying a user queue, this is the remote telephone number (if on has been set by the user) for this queue.

Can Take Additional Calls

This value is determined by comparing the number of call appearances with the number of active calls on the SIP station. The value

is TRUE if the device is a SIP station and the number of call appearances is greater than the number of active calls on the SIP station. Otherwise, the value is FALSE.

Exit Paths

Success

This step takes the Success exit path if the queue is alertable. See the tool overview for a list of these conditions.

OnPhone

This step takes the OnPhone exit path in the following circumstances:

Workgroup: If all members of the workgroup are unavailable and at least one workgroup member is on the phone.

User: If that user has an active call on his or her user queue.

Station: If that station has an active call or a call on hold (which would result from a switch hook flash operation).

UseStation

User Queues only: This step takes the UseStation exit path if the user is not logged in but has a ringable default workstation configured. This step only takes the UseStation exit path if the queue type is User and if the Success exit path is not taken. This would occur if the user is not logged in, but has one or more default workstation configured that do not have other users logged in to them.

Remote

User Queue only: This step takes the Remote exit path if the agent's status is set to Available, Forward and the agent has a remote telephone number.

Failure

This step takes the Failure exit path if no one is monitoring the queue and the queue is not a station queue. This step also takes the Failure exit path if the call object goes away.

Key Word Spotting

This Telephony tool enables CIC to start or stop monitoring a call for key words. If it identifies key words that are defined and associated with the specified workgroup in Interaction Administrator, it initiates notifications. In its most basic form, the tool stores the spotted key words and associates them with the call.

Inputs

Call Identifier

The unique identifier of the call.

Action

Start to begin monitoring for key words, or Stop to end monitoring of a call.

Workgroup Queue Id

The workgroup ID, which defines the list of key words to spot.

Language

The language associated with the call.

Exit Paths

Success

This tool takes the Success exit path if the operation is successful.

Failure

This tool takes the Failure exit path if the operation is not successful.

Listen From Call

This Telephony tool is similar to the Listen button on a CIC client toolbar. It allows one caller to listen to another user's call from an external call. This tool was written to enable Listen functionality for users not running a CIC client.

Inputs

Call Identifier

The InteractionId of the call to be listened to.

Call Identifier

The InteractionId of the listening call.

Continue Listening if call transferred?

Set this to True if you want to listen to the call after it is transferred. Set to False if you want to stop listening when the call is transferred.

Is this a supervisory listen?

Set to True or False to indicate whether or not this is a supervisory listen. This determines whether or not the person being listened to will be able to see that they are being listened to in the CID client. Supervisory listens are not detected by the person being listened to.

Continue listening call when target call finishes?

Set to True if you want the listening call to not disconnect when the target call finishes. For example, if you wanted to use the same listening call to listen to another call.

User Identifier

The UserID of the listening caller.

Exit Paths

Success

This path is taken if Listen functionality is successfully activated.

Failure

This path is taken if the operation fails. Listen could fail for various reasons, including using an incorrect InteractionId, or if the targeted caller disconnects before this tool is activated.

Listen From Station

This Telephony tool is similar to the Listen button on a CIC vlient toolbar. It allows one person at a specified station to listen to another user's call.

Inputs

Call Identifier

The InteractionId of the call to be listened to.

Station Queue Identifier

The ID of the station from which to listen to the call. (Examples: "Station:Gagle", "Station Queue:Gagle" or just "Gagle").

Continue Listening if call transferred?

Set this to True if you want to listen to the call after it is transferred. Set to False if you want to stop listening when the call is transferred.

Is this a supervisory listen?

Set to True or False to indicate whether or not this is a supervisory listen. This determines whether or not the person being listened to will be able to see that they are being listened to in a CIC client. Supervisory listens are not detected by the person being listened to.

Continue listening call when target call finishes?

Set to True if you want listening to continue when the target call finishes. For example, if you wanted to use the same listening call (returned as an output) to listen to another call.

Outputs

Call Identifier

The InteractionId of the listening call generated by this tool.

Exit Paths

Success

This path is taken if Listen functionality is successfully activated.

Failure

This path is taken if the operation fails. Listen could fail for various reasons, including using an incorrect InteractionId, or if the targeted caller disconnects before this tool is activated.

Log Message

This Telephony tool step writes a string to the Call Log attribute. Things that happen to a call are recorded and time stamped. For example, a call that leaves a voice mail message might have a log that looks something like this:

```
10:44:25: Call answered
10:44:25: System greeting
10:44:32: Success from lookup. Search Attribute = Queue Identifier Search Value = User
Queue:StephenS
10:44:32: Call transferred to User Queue:StephenS
10:44:32: Sending call to UserQueue Handler
10:44:32: Status: Unavailable
10:44:51: Send message requested
```

Each of these entries was written to the call log with a Log Message step. Use a [Get Call Log](#) step to retrieve the contents of a call log attribute. The contents can then be inserted in an email message of some other file.

When a call disconnects, CIC writes the contents of the Call Log attribute to the CallLog.log file.

Inputs

Call Identifier

The unique identifier for the call.

Call Log Identifier

The unique identifier that contains the name of the log you want to write a message to. The string identifier you specify here is the string identifier you must specify when extracting the contents with the Get Call Log tool.

Text of Log Message

The actual text that will appear in the log.

Exit Paths

Next

This step always takes the Next exit path.

Malicious Call Trace

This Telephony tool provides the ability to send a Malicious Call Trace (MCT) request on a SIP-based interaction. The request is received by the SIP endpoint and then processed. If the endpoint supports the MCT request and is using a Time Division Multiplexing (TDM) endpoint, it generates a Malicious Call ID (MCID) request on the TDM endpoint.

Note: Full use of this feature requires customizations to buttons and handlers in a CIC client. Contact PureConnect Customer Care for further information about this feature.

Inputs

Call Identifier

The call ID for the incoming call. This ID is used to look up the dialed number from which the call originated.

Initiate Recording?

If set to true, initiates a recording of the call. If set to false, the call is not recorded.

Exit Paths

Success

This path is taken if the operation is successful.

Failure

This path is taken if the operation fails.

Mute Interaction

This Telephony tool turns off the user's microphone. This allows the user to screen unwanted sounds or conversations from the caller without having to put the caller on hold.

Inputs

Call Identifier

The unique identifier for this call.

Mute Action ("Start" or "Stop")

Select **Start** to begin muting a call, or **Stop** to end muting and resume normal call activity.

Exit Paths

Success

The Success path is taken if the specified Mute Action is successfully applied.

Failure

The Failure path is taken if the specified Mute Action could not be applied. This will occur if the Call Identifier is incorrect.

Parallel Make Call

This Telephony tool fires the Parallel Make Call Outbound Call initiator and begins the Parallel Make Call process to make multiple outbound calls simultaneously from one station. One initiator is started for each call in the dial string list. This tool connects to the outbound call that connects first and abandons the other outbound calls.

Inputs

Line Groups

Specify a list of line groups (as configured in Interaction Administrator), or leave this parameter empty to use any available line group. In the default shipping handlers, the DialPlanEX subroutine returns a list of line groups to be used. You can pass that list in this parameter.

Station Queue Identifier

If this string is empty, the call is placed from the system queue. If the string identifies a station (for example, "Station:Gagle," "Station Queue:Gagle," or just "Gagle"), the call is placed from the named station and if there is a logged in user on that station, the user's queue.

Workgroup Queue Identifier

Identifies the workgroup from which this call originated. This parameter is useful for auto-dialers, such as the Interaction Dialer, when the ACD Server determines which agent a connected call should be sent to.

List of dial strings to be used

This list of dial strings parallels the list of line groups from the parameter above. When line group from position one is attempted, the dial string from position one is used, and so on. In the default shipping handlers, the DialPlanEX subroutine returns a list of dial strings that parallels the list of line groups. You can pass the list of dial strings from that subroutine into this parameter. A comma in a dial string causes a two-second pause, and any numbers after a "/" symbol are dialed after the call is connected.

Perform Call Analysis

Set this Boolean parameter to "True" if you want the handler to detect busy signals or whether the party did not pick up. Set to "False" if you are integrating with a PBX that does not generate the expected call progress audio. See [Call Analysis](#) for more information.

Notes: In-band voice messages (such as "This is AT&T, it is 4 P.M., where are you calling") can cause call analysis to fail. If this occurs, contact your long distance provider to have this message removed.

Note: If you turn off call analysis, calls made over analog, T1, and E1 lines will take the success exit path even if the call is busy. See [Call Analysis](#) for more information.

Include Answering Machine Detection in Call Analysis?

With this option selected, CIC attempts to detect an answering machine. If a machine is detected, this tool takes the Machine exit path. If this option is cleared, answering machines are not detected. See [call analysis](#) for more information on answering machine detection.

Silence Wait Timeout

The number of seconds of silence to elapse before taking the Machine exit path. This is useful if you want to ensure that the machine's greeting has finished playing before you play an audio file to the machine. This field is ignored if the **Include Answering Machine Detection** option is cleared.

No Answer Timeout

The number of seconds for the [Extended Place Call](#) tool to wait before taking the No Answer exit path. The default value is 30 seconds.

Parallel Make Call Timeout

The number of seconds to wait before taking the Timeout exit path.

Parallel Make Call Successes Desired

The minimum number of calls from the dial string list to be successfully completed before ending the Parallel Make Call subroutines. For example, if a user has a list of five dial strings but only needs one to connect, assigning this parameter a value of 1 will end the Parallel make Call process after the first call is connected.

Global Parameter

The parameter that is sent to each instance of the [Parallel Make Call Outbound Call](#) initiator.

Local Parameters

A list of local parameters parallel to the list of dial strings. I.e., the first local parameter is sent to the initiator for the first dial string, the second is sent for the second dial string, etc.

Calling Party Number

This parameter passes a string of digits to be displayed as ANI or Caller ID on the phone system that receives the call. If you do not specify a value in this parameter, the Telephony Services subsystem uses the number entered in the Phone Number field in the Line Configuration container in Interaction Administrator.

Note: This only works for ISDN.

Calling Party Name

This parameter passes a string to display the name of the caller on the phone system that receives the call. If you do not specify a value in this parameter, the Telephony Services subsystem uses the name associated with the number entered in the Phone Number field in the Line Configuration container in Interaction Administrator.

Note: This only works for ISDN.

Call for which this is a consulting call (optional)

If a call is a consulting call, the call ID for the call being consulted can be specified here.

Call Attribute Names

The names of any attributes to be assigned to the call as Queue Manager attributes before the Make Call is placed on TS.

Call Attribute Values

The values of any attributes named in the Call Attribute Names parameter above.

Outputs

List of Succeeded Calls

A list of calls from the original dialstring that were successfully made.

List of Failed Calls

A list of calls from the original dialstring that failed.

Exit Paths

Success

This path takes the Success exit path if it was able to compile the List of Succeeded Calls and the List of Failed Calls.

Timeout

This path is taken if the time set in the Parallel Make Call Timeout input expires.

Failure

This path is taken if the tool is unable to compile the List of Succeeded Calls and the List of Failed Calls.

Canceled

Parallel Make Call Failure

This Telephony tool flags an unsuccessful call to be placed on the List of Failed Calls generated with the [Parallel Make Call](#) tool.

Inputs

Call Identifier

The unique identifier for a call.

Exit Paths

Next

This tool always takes the Next exit path.

Park

This Telephony tool parks an interaction on the user queue you specify. The Park tool allows a handler to re-park an interaction after the original park operation has timed out. This tool allows handlers to differentiate between parked interactions timing out and held interactions timing out. For example, when a parked call timeout occurs, the Held Call Timer initiator is triggered (just as it is for held calls). If the call was parked, the ParkedFlag output parameter in the Initiator is set to true. The handler could then give the caller the option to re-park.

You can also use this tool to park an interaction or an Orbit queue. For more information, see the Queue Identifier input description.

Inputs

Call Identifier

The identifier for the interaction to park.

Queue Identifier

The user queue on which to park the call.

To park an interaction on an Orbit queue (extension 9999), specify:

Orbit Queue:9999

Exit Paths

Success

This tool takes the Success exit path if the call is successfully parked.

Failure

This tool can take the Failure exit path for several reasons. Failure can occur if the call disconnects, if the call is picked up by a user (and is no longer under the control of the handler), the call ID is no longer valid (if the call is deallocated), or there is a system resource limitation.

This tool also takes the failure path if you attempt to park the call on a non-user queue.

PickUp

This Telephony tool picks up a call that is on hold. Calls that are not in a state of On Hold cannot be picked up. This step does not pick up a call that has already been picked up by a CIC client user.

Inputs

Call Identifier

The unique identifier for call to be picked up.

Move Call to System Queue?

If this is set to True, the call is removed from any user or station queue which it is currently on and will be moved to the system queue. Set to False to leave the call on any user or station queues. In both cases the state of the call will be changed from **Held** to **Active**.

True: Claim control of this call

Set to True to place the call on hold and pass control of the call back to a handler, even if it was already picked up by a user. Set to False if you want this step to fail if a user has already picked up the call. The user will maintain control of the call.

Exit Paths

Success

This step takes the Success exit path if the call is successfully picked up.

Failure

This step takes the Failure exit path if the call was not on hold or had already been picked up by a user. This tool can take the Failure exit path for several other reasons. Failure can occur if the call disconnects, if the call is picked up by a user (and is no longer under the control of the handler), the call ID is no longer valid (if the call is deallocated), or there is a system resource limitation.

Place Call

This Telephony tool creates an outgoing call to an external or intercom number.

Inputs

Telephone Number

This is the number that will be called. In this parameter you can enter any string expression, including a literal telephone number, a queue ID, a variable that contains the value of a telephone number, or a complex expression built with the [Expression Editor Assistant](#). A comma causes a two-second pause, and any numbers after the "/" symbol are dialed after the call is connected.

Station Queue Identifier

If this string is empty, the call is placed from the system queue. If the string identifies a station ("Station:Gagle," "Station Queue:Gagle," or just "Gagle"), the call is placed from the named station. If there is a user logged in on that station, the call is placed from the user's queue.

Outputs

Call Identifier

The identifier for the new call.

Exit Paths

Success

If this step executes successfully, it takes the Success exit path.

Busy

If the number called is busy, it takes the Busy path.

Failure

If this step does not execute successfully, it takes the Failure exit path.

Play Audio File

This Telephony tool plays a .WAV file for a caller, and allows the caller to adjust volume and rewind or fast forward the file. This step does not play a prompt or CIC audio recording.

Notes: For best performance, all .WAV files should be in the format 8 kHz mono mu-law PCM. On-the-fly conversion is performed when the format of the audio in the .WAV file is anything other than 8 kHz mono mu-law PCM. If a .WAV file is going to be played often, the 8 kHz mono mu-law PCM format allows CIC to bypass Microsoft Audio Conversion Manager and play the audio with much less overhead.

Note: This tool does not support .MP3 audio files.

Inputs

Call Identifier

The unique identifier for a call.

Audio File Name

The unique name of the .WAV file. If you specify a path, make sure it is a fully qualified path indicating the server or drive letter. If you do not type a path, this tool uses the path stored in the Resource Path server parameter.

Key(s) to Increase Volume

The keypad keys a caller can press to increase volume.

Key(s) to Decrease Volume

The keypad keys a caller can press to decrease volume.

Key(s) to Skip Forward

The keypad keys a caller can press to advance the recording the number of seconds specified in Skip Amount.

Key(s) to Skip Backward

The keypad keys a caller can press to rewind the recording the number of seconds specified in Skip Amount.

Skip Amount (seconds)

The number of seconds to advance and rewind when a caller presses a Skip Forward or Skip Backward key.

Key(s) to Increase Playback Speed

The keypad keys a caller can press to speed up playback of the audio file.

Key(s) to Decrease Playback Speed

The keypad keys a caller can press to slow down playback of the audio file.

Immediate Mode

Set to True if you want the digits to be played when this step executes. Set to False if you want the digits to be queued with other digits played before and after so that several prompts are seamlessly played one after another.

Exit Paths

Next

This step always takes the Next exit path.

Play Digits

This Telephony tool plays one or more DTMF digits for a call. This tool could be used to send a series of digits representing a telephone number to a paging service, or any other time DTMF tones are required

Inputs

Call Identifier

The call to which the digits will be played.

Immediate Mode

Set to True if you want the digits to be played when this step executes. Set to False if you want the digits to be queued with other digits played before and after so that several prompts are played seamlessly, one after another.

Digits to be played

The digits to be played for the call.

Note: The following characters are supported in this field: 0-9, *, and #. Using a comma between digits adds a two-second pause when dialing. For more information, see the [_symbol](#) topic.

Exit Paths

Next

This step always takes the Next exit path.

Play Prompt

This Telephony tool step plays a CIC prompt recorded with the Prompt Editor. All prompts must be created by using the Edit button on the Inputs page, which opens the Prompt Editor. To play a prompt recorded in another handler, you must use the [Play Prompt Extended](#) tool. See [Record a prompt](#) for more information on creating prompts.

To make sure a prompt plays before the next tool step is executed, place an [Extended Get Key](#) step after the PlayPrompt and set the Timeout variable to a small decimal number (i.e., 0.01). Connect the next step from Timeout and leave the other exits unconnected. This will cause all audio prompts to be flushed before continuing.

Inputs

Call Identifier

The unique identifier for the call for which the prompt is played.

Language

The language in which the prompt will be played. You must have a recording in the language you choose. (Different language versions are defined when a prompt is created.)

Prompt Editor

Click the Edit button to open the Prompt Editor where you can select an existing prompt or create a new one. Whichever prompt you have selected when you click OK in the Prompt Editor is the prompt that is played when this step executes. The selected prompt's name also appears on the Inputs page of the properties notebook. The only prompts you can select from are those recorded for this handler or subroutine. If you want to select a prompt from another handler or subroutine, use the [Play Prompt Extended](#) tool.

Immediate Mode

Check this box if you want the prompt to be played when this step executes. Do not check this box if you want the prompt to be queued with other prompts played before and after so that several prompts are played seamlessly, one after another.

Exit Paths

Next

This step always takes the Next exit path.

Play Prompt Extended

Use this Telephony tool to play a prompt that was recorded in another handler. To use this tool, you will need to know the name of the prompt, the handler it was created in, and the language to be used. You may have to open the handler that contains the prompt you want to use in order to collect this information.

To make sure a prompt plays before the next tool step is executed, place an [Extended Get Key](#) step after the PlayPrompt and set the Timeout variable to a small decimal number (i.e., 0.01). Connect the next step from Timeout and leave the other exits unconnected. This will cause all audio prompts to be flushed before continuing.

If you want to record a new prompt, use the [Play Prompt tool](#).

Inputs

Call Identifier

The unique identifier for the call for which the prompt is played.

Handler Name

The name of the handler that contains the prompt to be played.

Language

The language in which the prompt will be played. You must have a recording in the language you choose. (Different language versions are defined when a prompt is created.)

Prompt Name

The name of the prompt to be played.

Immediate Mode

Check this box if you want the prompt to be played when this step executes. Do not check this box if you want the prompt to be queued with other prompts played before and after so that several prompts are seamlessly played one after another.

Exit Paths

Next

This step always takes the Next exit path.

Play Prompt Phrase

This Telephony tool allows you to specify a sequence of strings that define the structure of a prompt you want to play, and then substitute values at run-time to play the actual prompt. This eliminates the need to use multiple [Play Prompt](#) tools in order to play a sequence of prompts.

Inputs

Call Identifier

The unique identifier for the interaction.

Language

The language code of the prompt phrase to be played. If no language is specified, the language attribute of the call will be used. If no language is specified for the call either, the default language will be used.

Singular if true

This Boolean permits simple handler-based control over the common case of generating different prompt sequences for plural arguments than for singular ones (e.g., "one car" vs. "two cars"). Set to True to select the Singular Prompt Sequence. Set to False to select the Plural Prompt Sequence.

Prompt Strings

This tab allows the specification of zero or more Prompt Strings that may need to be substituted dynamically into the phrase. Typically, the entries on this tab are string literals or the names of string variables. A Prompt String consists of one or more prompt identifiers separated by white space. A prompt identifier can be scoped by a handler name or unscoped. A scoped prompt identifier consists of the handler name, a colon and the prompt ID (e.g., <HandlerName:PromptName>). All identifiers are case insensitive.

Variables

List of String Variables in the current handler.

Note: The string variables must be either a </tts:.....>, </file:.....> or a prompt in the form <PROMPT_NAME> for prompts contained within the handler, or <HANDLER:PROMPT_NAME> for prompts contained in another handler.

Parameters

The run-time substitution strings that are passed in at run-time to the sequence string.

Sequence Strings

A sequence string is a mechanism by which you arrange multiple prompts to be played at one time.

In the simplest Case, the Sequence String would be <%1>. This means there will be one substitution string from the Prompt Strings dialog below. The Prompt Strings dialog will either pass in a prompt, or a string variable which represents a prompt. However, the Prompt Sequence could be much more complex than that, so here's a more detailed example:

```
<Prompt_IVR:IVR_REMOTE_VM_MENU> <%1> <IVR_REMOTE_VM_EXIT> </file=thankyou.wav> </tts=mister smith>
```

This will play the following items in the order shown:

1. Prompt IVR_REMOTE_VM_MENU from the Prompt_IVR handler
2. %1 - substitution string
3. Prompt IVR_REMOTE_VM_EXIT from the current handler
4. A .WAV file: "thankyou.wav"
5. Text to Speech saying, "Mister Smith."

Single Sequence and Plural Sequence show what the sequence will be based on the Singular? parameter from the Inputs page.

This dialog lets you choose which sequence string to use for this phrase. The Handlers dialog lists the handlers you currently have open. Sequence Strings are resources just like prompts - they are resources of a handler. Simply highlight which handler you want to use and check the box of the sequence string you want to use.

Exit Paths

Next

This tool always takes the Next exit path.

Play Recording

This Telephony tool plays an audio recording. The only recordings that can be played with this step are CIC Audio Recordings that recorded with a [Record Audio](#) step. (CIC uses audio recordings to record voice mail.) You cannot play .WAV files or CIC Prompts using this step. .MP3 files are not supported.

To play a .WAV file for a listener, use a [Play Audio File](#) step. To play an CIC Prompt, use a [Play Prompt](#) step.

Inputs

Call Identifier

The unique identifier for this call.

Recording Identifier

The unique identifier for this recording.

Immediate Mode

Set to True if you want the digits to be played when this step executes. Set to False if you want the digits to be queued with other digits played before and after so that several prompts are played seamlessly, one after another.

Exit Paths

Next

This step always takes the Next exit path.

Play String

This Telephony tool reads the contents of a text string to a caller using the TTS engine.

Inputs

Call Identifier

The call to which the text string is played.

Text

The text that is played. This can also be the string variable.

Immediate Mode

Select this option if you want the text to be played when this step executes. Clear this option if you want the text to be queued with other prompts played before and after so that several prompts are played seamlessly, one after another.

Exit Paths

Next

This step always takes the Next exit path.

Related Topics

[Play String Extended](#)

Play String Extended

This Telephony tool reads the contents of a string to a caller using the text-to-speech (TTS) engine. This tool extends the options available over the [Play String](#) tool in that you can specify Voice Name, Volume, Speed, and other Optional Parameters.

For more information about installing, licensing, and configuring Interaction Text to Speech, see *CIC Text To Speech Engines Technical Reference* in the PureConnect Documentation Library.

Inputs

Call Identifier

The identifier for the call to which the string will be played.

Text

A text value or string variable containing the text to be played.

Voice Name

Name of voice to use to play the string.

Volume

An integer from zero to 100 representing the percentage of the maximum permitted volume at which the string will be played. That is, a value of 100 will play the recording at the maximum volume, whereas a value of 25 will play the recording at 25% of the maximum level.

Speed

This optional value is used to specify the approximate rate of speech. Since rates vary depending on what TTS engine is used, refer to the TTS provider documentation for more information.

Immediate Mode

Select this option if you want the text to be played when this step executes. Clear this option if you want the text to be queued with other prompts played before and after so that several prompts are played seamlessly, one after another.

Load Controller Timeout (milliseconds)

This parameter is no longer used.

Optional Parameters

Use this field to set any optional parameters to TTSServer.

One of the optional parameters you can specify is "NOXML." If the NOXML parameter is specified, the TTS engine ignores the < sign and does not attempt to process the text as a SAPI tag. If the NOXML parameter is not specified, the TTS engine attempts to process the < sign as the beginning of a SAPI tag.

To specify a name:value pair to control advanced MRCP properties, refer to [Using MRCP Tools](#).

To use ITTS even if you have SAPI or MRCP as the default TTS provider, specify "I3TTS" as the optional parameter.

To specify a name:value pair to control advanced ITTS properties, refer to in [Using ITTS Tools](#).

Exit Paths

Next

This step takes the Next exit path if the speech is played.

Rejected

This step takes the rejected exit path if the TTS subsystem does not have the resources to play the speech.

Play Text File

This Telephony tool plays the contents of a .TXT file using the Text-to-Speech engine.

Inputs

Call Identifier

The call to which the text string is played.

Text File Name (.txt)

The path to and name of the text file to be played. This can also be a string variable whose value is the path and name of a text file.

Immediate Mode

Select this option if you want the text to be played when this step executes. Clear this option if you want the text to be queued with other prompts played before and after so that several prompts are seamlessly played one after another.

Exit Paths

Next

This step always takes the Next exit path.

Related Topics

[Play Text File Extended](#)

Play Text File Extended

This Telephony tool plays the contents of a text file to a caller by using the Text-to-Speech (TTS) engine. This tool extends the options available over the [Play Text File](#) tool in that you can specify Voice Name, Volume, Speed, and Other Optional Parameters.

For more information about installing, licensing, and configuring Interaction Text to Speech, see *CIC Text To Speech Engines Technical Reference* in the PureConnect Documentation Library.

Inputs

Call Identifier

The call to which the text string is played.

Text File Name (.txt)

The path to and name of the text file to be played. This can also be a string variable whose value is the path and name of a text file.

Voice Name

Name of voice to use to play the string.

Volume

An integer from 0 to 100 representing the percentage of the maximum permitted volume at which the text file will be played. That is, a value of 100 will play the recording at the maximum volume, whereas a value of 25 will play the recording at 25% of the maximum level.

Speed

This optional value is used to specify the approximate rate of speech. Since rates vary depending on what TTS engine is used, refer to the TTS provider documentation for more information.

Immediate Mode

Select this option if you want the text to be played when this step executes. Clear this option if you want the text to be queued with other prompts played before and after so that several prompts are played seamlessly, one after another.

Load Controller Timeout (milliseconds)

This parameter is no longer used.

Optional Parameters

To specify a name:value pair to control advanced MRCP properties, refer to [Using MRCP Tools](#).

To use ITTS even if you have SAPI or MRCP as the default TTS provider, specify "I3TTS" as the optional parameter.

To specify a name:value pair to control advanced ITTS properties, refer to in [Using ITTS Tools](#).

Exit Paths

Next

This step takes the Next exit path if the text is successfully played.

Rejected

This step takes the rejected exit path if the TTS subsystem does not have the resources to play the speech.

Play Tone

This Telephony tool generates a tone for a connected call. For a list of standard tones, see [Tones](#).

Because the Play Tone tool does not have an Off Time parameter, you will not be able to generate Busy, Congestion, Reorder, or Ringback tones.

Note: Interaction Designer includes a prompt library named Prompt_Tones.ihd. This library contains tones from all around the world. You can use Interaction Designer to export any of the tones as .WAV files.

Inputs

Call Identifier

The unique identifier for a call.

Tone Frequency 1 in Hertz

The frequency of the tone you want to generate. By default this is set to 1100 Hz.

Tone Frequency 1 Amplitude in dB

The amplitude of the frequency you want to generate. By default this is set to -10.

Tone Frequency 2 in Hertz

The frequency of the second tone you want to generate. By default this is set to 0 Hz.

Tone Frequency 2 Amplitude in dB

The amplitude of the frequency you want to generate. By default this is set to -10.

Tone Duration

The number of seconds the tone plays. To convert milliseconds to seconds, divide your millisecond value by 1000. For example, if you want the tone to play for 500 milliseconds, you would type .5 in this parameter.

Exit Paths

Next

This step always takes the Next exit path.

Priority Set Attribute

This Telephony tool is for internal use only.

Priority Set Attributes

This Telephony tool is for internal use only.

Private

This Telephony tool flags a call as Private. Marking a call as Private:

- Prevents others from being able to use the Listen feature on that call.
- Stops an ongoing recording on the call.

Inputs

Call Identifier

The unique identifier of the interaction.

Make the call Private?

This Boolean determines whether or not to set the call as Private.

Exits

Next

This path is taken if the operation is successful.

Failure

This path is taken if the operation fails. Normally this will be because of an invalid call ID.

Query Conference Properties

This Telephony tool queries a conference call and returns information about the parties in that conference. Query Conference Properties returns the number of parties currently in the conference and three lists. Each list contains one element for each party in the conference.

Inputs

Conference Call Identifier

The unique identifier for the conference call created from the two objects on the Inputs page.

Outputs

Number of Current Parties

Number of parties presently in the conference.

Pipe "|" delimited list of each caller's time in conference, in seconds

A list containing the amount of time each party has been in the conference.

Pipe "|" delimited list of each conference party's CallId

A list containing the Call ID of each party involved in the conference.

Pipe "|" delimited list of each conference party's state

A list containing the Call State for each party involved in the conference.

Exit Paths

Success

This path is taken if the conference properties are successfully retrieved.

Failure

This path is taken if the operation fails.

Query Logged In Users

This Telephony tool queries a specified station queue and returns a list of all users currently logged in to that queue.

Inputs

Station Queue Identifier

The identifier for the station queue being queried.

Outputs

List of User IDs

List of strings containing the IDs for all users currently logged in to the specified station queue.

Exit Paths

Success

This path is taken if the list of logged in users is successfully returned.

Failure

This path is taken if the operation fails.

Query Media Subtype

This Telephony tool determines a call's media subtype, such as telephone calls or chat sessions. Use this tool as a branch point for processing different types of calls. Each supported media subtype has an associated exit path.

Inputs

Call Identifier

The identifier for the call you want to query.

Outputs

Media Subtype

A string that contains the media subtype (for example, fax or sms) for the call that was queried. If the media subtype is undefined this parameter contains an empty string.

Exit Paths

Undefined

If the interaction subtype is not a Fax or SMS session, this tool takes the Undefined exit path.

Custom

If the interaction subtype is not a Fax session, SMS session, or an undefined session, this tool takes the Custom exit path.

Fax

If the interaction subtype is an Fax session, this tool takes the Fax exit path.

SMS

If the interaction subtype is an SMS session, this tool takes the SMS exit path.

Failure

This path is taken if the operation fails.

Query Media Type

This Telephony tool determines an interaction's media type, such as telephone calls or chat sessions. Use this tool as a branch point for processing different types of interactions. Each supported media type has an associated exit path.

Inputs

Call Identifier

The identifier for the call you want to query.

Exit Paths

Telephone

If the call type is a telephone call, this tool takes the Telephone exit path.

Callback

If the call type is a callback, this tool takes the Callback exit path.

Chat

If the call type is a chat session, this tool takes the Chat exit path.

Web Collaboration

If the call type is a Web Collaboration session, this tool takes the Web Collaboration exit path.

Web

If the call type is a web session, this tool takes the Web exit path.

Generic Object

If the call type is a generic object, this tool takes the Generic Object exit path.

Email

If the call type is an email, this tool takes the Email exit path.

Monitor Object

If the call type is a monitor object, this tool takes the Monitor Object exit path.

Recorder Object

If the call type is a Recorder object, this tool takes the Recorder Object exit path.

SMS

If the interaction type is an SMS session, this tool takes the SMS exit path.

Conference

If the interaction is a conference call, this tool takes the Conference exit path.

Work Flow Work Item

If the interaction is a work item, the tool takes the Work Flow Work Item exit path.

Social Conversation

If the interaction is a social media conversation, then the tool takes the Social Media Conversation exit path.

Social Direct Message

If the interaction is a social media direct message, then the tool takes the Social Direct Message exit path.

Invalid

If the call identifier is no longer valid (because the call is no longer in the CIC), this tool takes the Invalid exit path.

Query Monitored Queues

This Telephony tool returns a list of queues monitored by a station. For example, your user and station queues are monitored after you start a CIC client. This tool would return your user queue as a queue that station was monitoring. This tool is called when a station goes off hook ([Station Off Hook](#)) to determine if a call should be connected to a station, or if dial tone should be played if no calls are waiting.

This tool sorts the queue items returned in this order: workstation, user, workgroup, other.

Inputs

Station Queue Identifier

The station queue to query.

Outputs

Queues actively monitored by this station

This list of string variable contains the list of queues monitored by this station. Pass this variable to the [Select Call](#) tool, or use the [InsertAtTail](#) tool to manually add additional monitored queues to the list.

Exit Paths

Next

This tool always takes the Next exit path.

Query Queue

This Telephony tool looks in a queue and returns information about the contents of that queue. Query Queue returns seven lists; each list has one element for each telephone or chat interaction in the queue.

Inputs

Queue Identifier

The name of the user and workgroup queue to be examined. This must be a fully scoped queue name such as "User Queue:StephenS" or "Workgroup Queue:Support".

Include Inactive Calls

Set this value to true to include inactive calls in the query. Set this value to false to exclude inactive calls from the query.

Outputs

Call Identifiers

A list of call identifiers for each interaction in the queue. In future releases, this may contain other types of identifiers for other types of queue items. These items are sorted chronologically with the oldest item as the first element in the list.

Remote Party Names

A list of remote party names that correspond to the queue items.

Remote Numbers

A list of telephone numbers associated with the queue items.

Status

The status of each item in the queue. These will be either Alerting, Active, Held, or Inactive.

States

A list of the states associated with each queue item. See [States](#) for a list of possible call states. These are state strings; they can also be any string assigned in a handler.

Creation Times

The time each queue item was created.

Time of last state change

The time each item in the queue last changed state.

Exit Paths

Success

This step takes the Success exit path if the information is successfully retrieved from the queue.

Failure

This step takes the Failure exit path if the queue does not exist.

Query Queue Type

This tool determines the type of queue (User, Workgroup, or Station).

Inputs

Queue Identifier

The unique identifier for a queue. This is the queue whose type will be determined.

Outputs

Non-Scoped Queue Name

Non-Scoped Queue Name refers to the name of the queue object without the queue type; for example, a phone is referred to as "User Queue:MikeG" (a scoped queue name) and the Non-Scoped name is just "MikeG."

Exit Paths

Station

This exit path is taken if the queue type is Station.

Workgroup

This exit path is taken if the queue type is Workgroup.

User

This exit path is taken if the queue type is User.

Failure

This exit path is taken if the queue type is not of the type Station, Workgroup, or User.

Query Statuses for User

This Telephony tool returns a list of statuses for which the user has rights.

Inputs

User

The user ID.

Outputs

List of Statuses

The list of statuses that are available for the specified user.

Exit Paths

Next

If the user is found, this step takes the Next exit path.

Failure

Fails if the user is not found.

Query User Status

This Telephony tool returns a user's current status and the DateTime value that user set.

Inputs

User Identifier

The identifier for the user whose status you want to query. You can format this value in the following ways: "User Queue:StephenS," "User:Stephens" or "StephenS."

Outputs

Status Message Name

A string containing the status to which the user is set.

Date and Time for status

A datetime variable containing the date and time for the status. For example, if the user has set his or her status to "On Vacation until 5/5/2000," this variable would have a value. If the user has not set a date, the value is empty.

Is Date in StatusDateTime set?

A boolean value that is "true" if the date is set for the status and "false" if the date is not set.

Is Time in StatusDateTime set?

A boolean value that is "true" if the time is set for the status and "false" if the time is not set.

Is User Logged in?

A boolean value that is "true" if the user is logged in and "false" if the user is not logged in.

Is Status 'Do not Disturb'?

A boolean value that is "true" if the status is a 'do not disturb' status and "false" if the status is not.

Is Status ACD?

A boolean value that is "true" if the status is an ACD status and "false" if the status is not.

Is Status Forward?

A boolean value that is "true" if the status is a forwarding status and "false" if the status is not.

Is Status On Phone?

A boolean value that is "true" if the status is a 'on the phone' status and "false" if the status is not.

Exit Paths

Next

This tool takes the Next exit path if the status is returned.

Failure

This tool takes the Failure exit path if the user identifier is invalid.

Record Audio

This Telephony tool step records a caller's message and stores it temporarily (in the directory specified in the Recording Path server parameter) as an CIC Audio Recording for playback in a [Play Recording](#) step or for a conversion to a .WAV in a [Send Voice Mail](#) step. The .WAV file is discarded after a handler containing this step runs. If you need to save the .WAV file, use the [Record File](#) tool.

This step does not convert the Audio Recording to .WAV format. Conversions to .WAV are performed in the [Send Voice Mail](#) step.

If a caller presses a key on the telephone keypad, the value of that key is stored as an output value and this handler takes the success exit path.

Note: Record Audio fails if the call it records is not in a Connected state. If you are building a handler that uses the Record Audio step, you can use a Pickup step to change the state from Notifying or Offering or On Hold to Connected.

Inputs

Call Identifier

The unique identifier for the call to be recorded.

Valid Keys

These are keys that can be pressed by the user and accepted for this step. Use this parameter to set up which key a user can press to indicate that he or she has finished recording.

Note: A user can also indicate that a message is complete by hanging up.

Escape Keys

The key a caller can press to take the Escape exit path.

Tone Detection Frequency 1 in Hertz

1100 To disable Fax tone detection, set this to 0.

Permissible deviation in Tone Detection Frequency 1

50

Tone Detection Frequency 2 in Hertz

0

Permissible deviation in Tone Detection Frequency 2

50

Time in seconds which tone must be on

0.200000

Permissible deviation in tone on time

-0.200000

Time in seconds which tone must be off

0.000000

Permissible deviation in tone off time

0.000000

Number of tone on/off repetitions required for detection

0

Timeout (seconds)

The maximum length of the recording, and the number of seconds this step will record before taking the Success exit path. The value of this parameter is set to 600 seconds (10 minutes) by default. Setting the Timeout to 0 or a negative number causes this tool to record indefinitely. Specifying any value greater than 600 results in the default of 600 seconds.

Note: A server parameter called Voicemail Maximum Duration is available to pass a value to this tool step and override the default value. For more information, see the "Packaged Server Parameters" topic in the *Interaction Administrator Help*.

Silence Compression Enabled

Select this checkbox if you want any long periods of silence to be removed from the Audio Recording. Select this checkbox to remove long periods of silence at the beginning or end of the recording. This is supported by the media server and defaults to 2 seconds.

Final Silence (seconds)

The number of seconds of silence at the end of an utterance that will terminate the recording (defaults to 10). When set to 0, this parameter indicates that the Interaction Administrator system line setting for voicemail silence detection is used. When set to a number less than 0, it is disabled.

Mime Type

The audio format to use for recording the audio (when blank, defaults to the format specified on the media server). Available audio formats include:

- Audio/L16: Linear 16-bit
- Audio/PCMU: muLaw
- Audio/PCMA: aLaw
- Audio/G726-32: G726 - 32bit
- Audio/GSM: GSM
- Audio/x-truespeech: DSP TrueSpeech

Automatic Level Control

Indicates whether Automatic Level Control should be applied to the audio (defaults to false).

Outputs

Recording Identifier

The variable that identifies this recording. The recording is stored in the directory specified in the Recording Path server parameter.

Length of recording (seconds)

The variable that contains the amount of audio that was recorded.

Keys

The value of a pressed key. It can only be one of the keys defined as Valid or Escape key(s). If you allow a person recording audio to perform actions on that audio by pressing special keys, the key pressed is recorded in this variable.

Exit Paths

Success

This step takes the Success exit path if a non-escape key is pressed or if the recording is longer than the number of seconds specified in the Timeout parameter.

Escape

This exit path is taken if the person recording the audio pressed one of the escape keys.

Cancelled

Call ownership changed. This can occur if a user picks up a call while voice mail is being recorded.

Failure

If the call ID is no longer valid, this tool takes the Failure exit path. This can occur if the caller hangs up before this step executes. If the caller hangs up after recording has started, this tool takes the Success exit path.

Tone

This step takes the Tone exit path if a tone is detected.

Record Call

This Telephony tool records a telephone or chat interaction. Use this tool when you want to record a call for the lifetime of that call, as opposed to recording some portion of the call while it is in a certain handler like Voice Mail.

This tool uses an existing recording created by using the Record button in a CIC client, if one exists. When creating a new recording, Telephony Services uses a generic system ID, rather than the local user ID.

At the time the recording ends, the Eic_RecordFileName attribute is set to the file name and path for the storage location of the recording. Inputs

Call Identifier

The identifier for the recorded telephone or chat interaction.

Record Action

Type either Start, Stop, or Pause. After pausing, a Start action will continue recording.

Continue recording if call transferred?

Set to True to continue the recording of a call even if the call is transferred.

Recording Requested By

The user requesting the recording.

Is this a supervisory record?

Set to True to indicate that the user requesting the recording has supervisory status.

Mime Type

The audio format to use for recording the audio (when blank, defaults to the format specified on the media server). Available audio formats include:

- Audio/L16: Linear 16-bit
- Audio/PCMU: muLaw
- Audio/PCMA: aLaw
- Audio/G726-32: G726 - 32bit
- Audio/GSM: GSM
- Audio/x-truespeech: DSP TrueSpeech

Automatic Level Control

Indicates whether Automatic Level Control should be applied to the audio (defaults to false).

Outputs

Recording Call Identifier

The unique identifier for the recording interaction that is handling the recording of the call.

Exit Paths

Success

This step exits along the success path if the call is successfully recorded.

Failure

This tool can take the Failure exit path for several reasons. Failure can occur if the call disconnects, if the call is picked up by a user (and is no longer under the control of the handler), the call ID is no longer valid (if the call is deallocated), or system resource limitation.

Record File

This Telephony tool records telephone conversation and saves it in the form of a .WAV file. This tool also accepts a key press from a telephone keypad. As soon as any key is pressed, this tool exits and the recording is saved. Any time this tool exits, the recording is saved. This tool is used in the SystemIVRRecordPrompt handler for CIC client users to record their personal prompts. You can use this tool any time you want to record a call as a .WAV file.

This tool will overwrite any .WAV files with the same name in the same location.

This tool is different from [Record Audio](#) because Record File saves the .WAV file, where Record Audio deletes its recording when the handler ends.

Inputs

Call Identifier

The identifier for the recorded call.

Valid Keys

The keys that are acceptable as input for this tool. If a user presses one of these keys, this tool stops recording and takes the Successful exit path.

Escape Keys

The key press that causes this tool to take the Escape exit path. If a user presses one of these keys, this tool stops recording and takes the Escape exit path.

Timeout (seconds)

The maximum allowable recording length before taking the Success path.

Note: You can either specify a maximum timeout of up to 600 seconds, or set the value to -1 to specify no limit.

Audio File Name (.wav)

The name to use when saving the .WAV file, including the .WAV extension. If there is already a file with that name, it is overwritten when this step executes. If you specify a path, make sure it is a fully qualified path indicating server or drive letter. If you do not type a path, this tool uses the path stored in the Recording Path server parameter.

If the filename or path is invalid, this tool takes the Failure exit path.

Silence Compression Enabled

Set to True to remove long periods of silence in the recording.

Append if File Already Exists

When this is set to True, any additional recordings to the same file name are appended to the original file. If set to False, any additional recordings to the same file overwrite the original file.

Insert Tone if Appending

If this option is set to True, a tone is inserted before each appendage. If set to False, a tone is not inserted before each appendage.

Final Silence (seconds)

The number of seconds of silence at the end of an utterance that will terminate the recording (defaults to 10). When set to 0, this parameter indicates that the Interaction Administrator system line setting for voicemail silence detection is used. When set to a number less than 0, it is disabled.

Mime Type

The audio format to use for recording the audio (when blank, defaults to the format specified on the media server). Available audio formats include:

- Audio/L16: Linear 16-bit
- Audio/PCMU: muLaw
- Audio/PCMA: aLaw
- Audio/G726-32: G726 - 32bit
- Audio/GSM: GSM
- Audio/x-truespeech: DSP TrueSpeech

Automatic Level Control

Indicates whether Automatic Level Control should be applied to the audio (defaults to false).

Outputs

Length of Recording

The duration of the recording in seconds.

Keys

Any keys the caller has pressed. Once a valid or escape key has been pressed, this tool exits.

Exit Paths

Success

This step takes the Success path when one of the valid keys is pressed or if the number of seconds in the Timeout parameter is exceeded.

Escape

This step takes the Escape path if an escape key is pressed.

Cancelled

This step takes the Cancelled path if someone else takes control of this call object.

Failure

This tool can take the Failure exit path for several reasons. Failure can occur if the call is already disconnected, if the call is picked up by a user (and is no longer under the control of the handler), the call ID is no longer valid (if the call is deallocated), or system resource limitation.

Tone

This step takes the Tone exit path if a tone is detected.

Record String

This Telephony tool reads the contents of a string into an audio file (.WAV) using the Text-to-Speech engine.

Inputs

Text

The text that is recorded. This can also be the string variable.

Audio File Name (.WAV)

The path to and name of the .WAV file that is to be created. If you do not include any path information, the file will be generated in the directory specified by the "TTS Audio File Directory" server parameter. If no directory is specified, it defaults to whatever the Win32 API function GetTempPath uses, which typically returns "C:\TEMP" when called from a normal application and "C:\WINNT" when called from a service.

Outputs

Length of recording

An integer approximation of the playback length. When the server is running under a reasonable load, the record time should be 2-8 times quicker than the playback time.

Exit Paths

Success

This step takes the Success exit path if the .WAV file is created.

Failure

This step takes the Failure exit path if one of the following occurs:

- The server has no TTS engine installed.
- The audio file already exists and is read-only.
- The process does not have write permission in the directory specified for the audio file.
- The string was empty or contained nothing that could be converted into "phonemes" (i.e. no audio was generated).

Related Topics

[Record String Extended](#)

Record String Extended

This Telephony tool reads the contents of a string into an audio file (.WAV) using the Text-to-Speech (TTS) engine. This tool extends the options available over the [Record String](#) tool in that you can specify Voice Name, Speed, and other Optional Parameters.

For more information about installing, licensing, and configuring Interaction Text to Speech, see *CIC Text To Speech Engines Technical Reference* in the PureConnect Documentation Library.

Inputs

Text

The text that is recorded. This can also be the string variable.

Audio File Name (.WAV)

The path to and name of the .WAV file that is to be created. If you do not include any path information, the file will be generated in the directory specified by the "TTS Audio File Directory" server parameter. If no directory is specified, it defaults to whatever the Win32 API function GetTempPath uses, which typically returns "C:\TEMP" when called from a normal application and "C:\WINNT" when called from a service.

Voice Name

Name of voice to use to record the string.

Speed

This optional value is used to specify the approximate rate of speech. Since rates vary depending on what TTS engine is used, refer to the TTS provider documentation for more information.

Optional Parameters

To specify a name:value pair to control advanced MRCP properties, refer to [Using MRCP Tools](#).

To use ITTS even if you have SAPI or MRCP as the default TTS provider, specify "I3TTS" as the optional parameter.

To specify a name:value pair to control advanced ITTS properties, refer to in [Using ITTS Tools](#).

Outputs

Length of recording

An integer approximation of the playback length.

Exit Paths

Success

This step takes the Success exit path if the .WAV file is created.

Rejected

This step takes the Rejected exit path if one of the following occurs:

- The server has no TTS engine installed.
- The audio file already exists and is read-only.
- The process does not have write permission in the directory specified for the audio file.
- The string was empty or contained nothing that could be converted into "phonemes" (i.e. no audio was generated).

Record Text File

This Telephony tool reads the contents of a text file into an audio file (.WAV) using the Text-to-Speech (TTS) engine.

Inputs

Text File Name (.txt)

The name of the file containing the text that is recorded.

Audio File Name (.WAV)

The path to and name of the .WAV file that is to be created. If you do not include any path information, the file will be generated in the directory specified by the "TTS Audio File Directory" server parameter. If no directory is specified, it defaults to whatever the Win32 API function GetTempPath uses, which typically returns "C:\TEMP" when called from a normal application and "C:\WINNT" when called from a service.

Outputs

Length of recording

The length of the recording in seconds.

Exit Paths

Success

This step takes the Success exit path if the .WAV file is created.

Failure

This step takes the Failure exit path if one of the following occurs:

- The server has no TTS engine installed.
- The audio file already exists and is read-only.
- The process does not have write permission in the directory specified for the audio file.
- The string was empty or contained nothing that could be converted into "phonemes" (i.e. no audio was generated).
- The specified text file could not be accessed.

Related Topics

[Record Text File Extended](#)

Record Text File Extended

This Telephony tool reads the contents of a text file into an audio file (.WAV) using the Text-to-Speech (TTS) engine. This tool extends the options available over the [Record Text File](#) tool in that you can specify Voice Name, Speed, and other Optional Parameters.

For more information about installing, licensing, and configuring Interaction Text to Speech, see *CIC Text To Speech Engines Technical Reference* in the PureConnect Documentation Library.

Inputs

Text File Name

The text that is recorded. This can also be the string variable.

Audio File Name (.WAV)

The path to and name of the .WAV file that is to be created. If you do not include any path information, the file will be generated in the directory specified by the "TTS Audio File Directory" server parameter. If no directory is specified, it defaults to whatever the Win32 API function GetTempPath uses, which typically returns "C:\TEMP" when called from a normal application and "C:\WINNT" when called from a service.

Voice Name

Name of voice to use to record the string.

Note: You can specify a language in Interaction Administrator when defining a voice. For more information, refer to the Interaction Administrator help.

Speed

This optional value is used to specify the approximate rate of speech. Since rates vary depending on what TTS engine is used, refer to the TTS provider documentation for more information.

Optional Parameters

To specify a name:value pair to control advanced MRCP properties, refer to [Using MRCP Tools](#).

To use ITTS even if you have SAPI or MRCP as the default TTS provider, specify "I3TTS" as the optional parameter.

To specify a name:value pair to control advanced ITTS properties, refer to in [Using ITTS Tools](#).

Outputs

Length of recording

An integer approximation of the playback length.

Exit Paths

Success

This step takes the Success exit path if the .WAV file is created.

Rejected

This step takes the Rejected exit path if one of the following occurs:

- The server has no TTS engine installed.
- The audio file already exists and is read-only.
- The process does not have write permission in the directory specified for the audio file.
- The text file was empty or contained nothing that could be converted into "phonemes" (i.e. no audio was generated).

Reload Station

This Telephony tool sends a SIP message (check-sync) to cause the phone to reload/restart.

Inputs

Station Queue Identifier

Unique identifier for the station queue.

Exit Paths

Success

This exit path is taken if the SIP message is sent to reload/restart the phone.

Failure

This exit path is taken if the SIP message is not sent.

Remove Party

This Telephony tool is used to remove a specified call from a conference without disconnecting the call.

Inputs

Call Identifier

The unique identifier for a call. In this instance, a call to a party you want to conference.

Exit Paths

Success

This path is taken if the call is successfully removed from the conference.

Failure

This path is taken if the operation fails.

Reset Password

This Telephony tool resets an CIC user's password to a randomly generated password.

Note: As the handler author, you are responsible for authenticating the user via the TUI and delivering the new password to the user.

Inputs

User Identifier

The ID of the user to reset.

Numeric Only

If true, a numeric password is generated. If false, the password will be a combination of alpha and numeric characters.

Outputs

New Password

The password to which the account was reset.

Exit Paths

Success

The Success exit path is taken if the password is successfully generated.

Invalid User

This exit path is taken if the user associated with the ID is not found.

Failure

The Failure exit path is taken if the operation fails.

Secure Session Begin

This Telephony tool starts a secure session and starts the [Secure Input Initiator](#).

The default name of the handler called by the Secure Session Begin tool is SecureInputSub. Create the SecureInputSub handler to use the Secure Input Initiator. To call a different handler, add I3_RESERVED_IVR_HANDLER_NAME to the Auxiliary Data Names parameter and add the handler name to the Auxiliary Data Value parameter at the same index.

Inputs

Call Identifier

The unique identifier for the call.

Auxiliary Data Names

A list of labels.

Auxiliary Data Values

A list of names.

Outputs

Transaction Id

The unique identifier for the transaction.

Exit Paths

Success

The tool takes this path if the operation is successful.

Failure

The tool takes this path if the operation fails.

Secure Session End

This Telephony tool ends a secure session and reconnects the audio paths.

Inputs

Call Identifier

The unique identifier for the call.

Exit Paths

Success

The tool takes this path if the transaction is successful.

Failure

The tool takes this path if the transaction fails.

Secure Session Get Key

This Telephony tool is similar to [Extended Get Key](#) except that Secure Session Get Key allows additional checks within Telephony Services and provides secure storage within the Telephony Services subsystem. To prevent logging of secure data, the tool does not return any data to handlers.

Inputs

Call Identifier

The unique identifier for a call.

Name

A label used when sending data for a validation step.

Regular Expression

A regular expression in the Boost.Regex format. The tool performs a check against this regular expression.

Additional Checks

An input string that contains "Luhn". This parameter currently supports only the Luhn algorithm check. For more information, search for "Luhn algorithm" on the Internet.

Valid Keys

0,1,2,3,4,5,6,7,8,9 are the keys a caller may enter. You can change these keys using the [Expression Editor Assistant](#).

Escape Keys

"*" is the escape key. When a caller presses the escape key, this step stops accepting caller input and takes the Escape exit path. Change these keys using the Expression Editor Assistant.

Termination Keys

"#" is the termination key for this step. This step immediately stops waiting for key input when this key is pressed and exits through the Success exit path. If the caller presses the termination key before entering any other keys, this tool takes the Success exit path. The termination key appears in the Digits output key if it was included in the Valid Keys input. Otherwise, the Digits output variable is empty.

Maximum number of Keys

The maximum number of keys this step accepts before taking the Successful exit path.

Maximum Inter-Digit Delay

The number of seconds to allow between key presses before this step takes the Success exit path. The default is 2.5 seconds. The Inter-Digit delay is not counted until the caller presses the first key. If the inter-digit delay expires, the tool will exit via the Success exit path with an output of whatever digits the caller pressed up to that point.

Note for the following Tone Detection parameters: The default values for the next 8 settings recognize a standard fax tone. In almost all cases, you do not need to change the default settings. If you want to detect a tone other than a fax device, refer to the documentation for the device generating the tone. Also see the topic on [Tones](#).

Tone Detection Frequency 1 in Hertz

1100

To disable Fax tone detection, set this to 0.

Permissible deviation in Tone Detection Frequency 1

50

Tone Detection Frequency 2 in Hertz

Use 0 to disable.

Permissible deviation in Tone Detection Frequency 2 (in Hertz)

50

Time in seconds which tone must be on

Amount of time that the tone must last.

Permissible deviation in tone on time

The number of seconds that the tone on time can deviate from the value in the **Time in seconds which tone must be on** setting.

Time in seconds which tone must be off

Amount of time that the tone must last.

Permissible deviation in tone off time

The number of seconds that the **tone off** time can deviate from the value in the **Time in seconds which tone must be off** setting.

Number of tone on/off repetitions required for detection

0

Timeout (seconds)

The total number of seconds after audio is played for any keys to be entered before this tool takes the Timeout exit path. The default for this parameter is 20 seconds. If you are using this Extended Get Key step simply to flush any queued audio, set this value to 0 seconds. You should not set this value to zero unless you are flushing audio. The maximum value for this tool is 600 seconds. Any value specified greater than 600 will cause this tool to wait for 600 seconds.

This tool accepts a caller's key presses while the audio is being played.

Attribute to monitor as a termination event

This optional parameter specifies the name of an attribute to be monitored during the operation of this tool. If the value of this attribute changes during the Extended Get Key operation, the tool will exit taking the Attribute exit path.

Interaction ID for the attribute monitor

This optional parameter specified the interaction ID associated with the attribute to monitor.

Repeat queued plays if audio interruption is encountered?

If set to true, queued plays are repeated if audio is interrupted (a voice mail greeting is detected).

Outputs

Failed Input Checks

A string that specifies regex or Luhn to indicate failure details on Failure exit path.

Exit Paths

Success

This step takes the Success exit path if it does not take one of the exits below. If the caller pressed the termination key (#) without entering any digits, this tool takes the Success exit path and the Digits output variable is empty.

Escape

This step takes the Escape exit path if the caller presses an escape key.

Tone

This step takes the Tone exit path if it detects the tone specified by the Tone parameters on the Inputs pages.

Timeout

This step takes the Timeout exit path if no keys are pressed within the amount of time allotted in the Timeout parameter on the Inputs page.

Failure

This tool can take the Failure exit path for several reasons. Failure can occur if the call disconnects, if the call is picked up by a user (and is no longer under the control of the handler), the call ID is no longer valid (if the call is deallocated), or system resource limitation.

Attribute

This path is taken if an attribute that has been set to be monitored changes during the operation of this tool.

Secure Session Info Insert

This Telephony tool attaches data to use in a validation request.

Note: This data is for convenience and is not secure.

Inputs

Call Identifier

The unique identifier for the call.

Name

The label to use when sending data to the [Secure Session Info Validate](#) tool step.

Value

Exit Paths

Success

The tool takes this path if the operation is successful.

Failure

The tool takes this path if the operation fails.

Secure Session Info Validate

This Telephony tool sends securely stored data from Telephony Services, as well as auxiliary data and data from the Secure Session Info Inserted tool, to a web application that calls a third-party credit card validation or other secure operation.

Note: The web application is written by the customer. To download an example handler, click **Sample Handler for Secure IVR** from the Utilities and Downloads page at <https://help.genesys.com/utilities-and-downloads.html>.

Inputs

Call Identifier

The unique identifier for the call.

Web-Service Uri

The URI of the web application, such as: <https://internalweb.inin.com/webpage.asp>

Factory Label

Indicates which web request factory to use. Default value is "default".

Outputs

Approval Value

A numeric result code.

```
eTELEPHONY_INFOVALIDATE_FAILED=0  
eTELEPHONY_INFOVALIDATE_APPROVED=1  
eTELEPHONY_INFOVALIDATE_DECLINED=2  
eTELEPHONY_INFOVALIDATE_OTHER=3
```

Approval Result

A string result message from the web application.

Exit Paths

Validate Failed

The tool takes this exit path if the web application fails to validate the secure data.

Validate Approved

The tool takes this exit path if the secure data is successfully validated.

Validate Declined

The tool takes this exit path if web application declines the use of the secure data.

Validate Other

The tool takes this exit path if the web application returns any other result.

Select Call

This Telephony tool searches a list of queues serially to determine if one of those queues contains a call that should be assigned to the station. The call ID for an alerting or held call in one of the listed queues is returned. Any remaining conditions in the list are ignored. If multiple calls in the queue satisfy the criteria, the one with the longest time in that has been alerting or on hold the longest is used. This tool is typically used after CIC detects a station has gone off-hook, triggering the [Station Off Hook](#) initiator. If none of the following conditions are met, dial tone is played.

Note: This tool does not take a user's status into consideration. If your status is set to Do Not Disturb, you will still receive an alerting call or a call on hold when you pick up your station.

Condition	Action taken
Station queue has an alerting call	Returns that call ID.
Station queue has a call on hold	Returns that call ID.
Any user queue being actively monitored by the station has an alerting call	Returns that call ID.
Any user queue being actively monitored by the station has a held call	Returns that call ID.
Any workgroup queue being actively monitored by the station has an alerting call	Returns that call ID.
Any workgroup queue being actively monitored by the station has a held call	Returns that call ID.
No-one is logged in at this station and any user queue for which the station is the default workstation has an alerting call	Returns that call ID.
No-one is logged in at this station and any user queue for which the station is the default workstation has a held call	Returns that call ID.
	Dialtone for an outgoing call.

Inputs

Queues to Search

The list of queues to search. This list is typically, but does not have to be, output from the [Query Monitored Queues](#) tool.

Select on-hold calls from Station Queue only?

Set to True if you do not want to pick up held calls from the workgroup queue. Note that calls in the user queue will also appear in the station queue and thus still be available when this option is selected.

Outputs

Call Identifier

The ID for the call to connect to the station.

Exit Paths

Success

This tool takes the Success exit path if the Queues to Search are valid queues.

Failure

This tool takes the Failure exit path if Queues to Search contains invalid queue identifiers, or if none of the searched queues contain alerting or held calls.

Send ADSI String

This Telephony tool sends two strings to a station queue to be displayed on an ADSI (Analog Display Services Interface) capable telephone. While the labels for this tool indicate that you should send a telephone number and name to display, you can send any string. The string is only sent when the phone is not connected to another CIC audio source (such as receiving dial tone, listening to voicemail, connected to a call, etc.).

To clear the text displayed on the phone, call another instance of this tool with empty Calling Number and Calling Name strings.

The station to which the strings are sent must be configured with the ADSI option selected. See ADSI in the Interaction Administrator online help for more information on configuring a station to receive ADSI information.

Inputs

Station Queue Identifier

The station on which to display the strings.

Calling Number to be Displayed

The first string to be displayed on the ADSI telephone.

Calling Name to be Displayed

The second string to be displayed on the ADSI telephone.

Exit Paths

Next

This tool always takes the Next exit path.

Set Attribute

This Telephony tool step assigns a new attribute to a telephone call or chat session, and assigns the value of that newly assigned object attribute. When the object disappears, so do the attributes. Object attributes can be thought of as temporary storage, and only last as long as the object. Object Attributes are not the same attributes in Directory Services Keys. A call log is an example of one of these attributes. When the call goes away, so does the log. For a list of the existing call object attributes, see the *Interaction Attributes Technical Reference* in the PureConnect Documentation Library.

Only strings can be stored as the value of an attribute.

Note: This tool cannot set a value for an attribute in a Directory Services key. To set a value in a Directory Services Key, use the [PutDsAttr](#) and [PutDsAttrS](#) tools.

Inputs

Call Identifier

The unique identifier for a call.

Call Attribute Name

The name of the attribute to be set.

Call Attribute Value

The value of the attribute to be set.

Exit Paths

Next

This step always takes the Next exit path.

Set Attributes

This Telephony tool assigns one or more values to a list of one or more call attributes. The attributes are created if they do not exist. This tool works for telephone calls and chat sessions.

Inputs

Call Identifier

The identifier for the call to to which you want to assign the attribute values.

Call Attribute Names

The list of attributes for which you want to create or set values.

Call Attribute Values

A parallel list of values. For example, if the List of Attribute Names contains two items, then the List of Attribute names must also contain two values at the same positions in the list of strings.

Exit Paths

Next

This step always takes the Next exit path.

Set Billing Rate

This Telephony tool should only be used if you are using AT&T's Vari-A-Bill service. Set Billing Rate instructs AT&T to charge a customer a specific amount for a specific call. For more information on how Vari-A-Bill works, see the Vari-A-Bill User Guide you received from your Vari-A-Bill vendor.

Contact your AT&T Technical Sales Specialist to arrange testing of any Vari-A-Bill functionality you create in handlers.

Vari-A-Bill is optional on each line and must be activated in the Line Configuration container in Interaction Administrator.

Inputs

Call Identifier

The call that will have a billing rate set.

Billing Rate Type

These are the types of billing rates that can be charged.

Note: You can only use one of these types for a specific call, although you may send other messages of the same type. AT&T uses the final rating as the basis for the billing. The only exception to this is that Premium Charge and Premium Credit can be used interchangeably on the same call with the other three options.

ISDN_FREE_CALL - There is no charge for the entire call. This can be used at any time during the call.

ISDN_NEW_RATE - A new per minute rate is applied to the remainder of the call. Minutes incurred prior to applying New_Rate are billed at the original provisioned rate.

ISDN_FLAT_RATE - A flat charge is applied to the remainder of the call. The flat charge is added to the previously incurred per-minute charges.

ISDN_PREM_CHARGE - An additional flat charge is added on top of the provisioned charges that continue to accrue.

ISDN_PREM_CREDIT - A negative flat rate is applied to the call. Previously incurred charges continue to accrue.

Rate Change (cents)

The number of cents to charge. If this is ISDN_NEW_RATE rate, it is the cents per minute. If this is ISDN_FLAT_RATE, ISDN_PREM_CHARGE, or ISDN_PREM_CREDIT, it is the total amount to charge or credit.

Exit Paths

Success

This step takes the Success exit path if the rate is set.

Failure

This tool can take the Failure exit path for several reasons. Failure can occur if the call disconnects, if the call is picked up by a user (and is no longer under the control of the handler), the call ID is no longer valid (if the call is deallocated), or system resource limitation.

Set Call State

This Telephony tool step changes the state of a call. Set Call State could be used to indicate to CIC client users that a call is in voice mail or sending/receiving a fax. While calls delivering voice mail or a fax are actually in a state of Connected, CIC client users will see "Voice Mail" or "FAX."

See [States](#) for information on state strings.

Inputs

Call Identifier

The unique identifier for the call.

New Call State

The state to which the call is set.

Voicemail recording will be performed

Set to True if the call will be transferred to voice mail. This prepares the call for recording.

Call should be moved to the system queue

Set to True if the call should be moved to System queue.

True: Claim control of this call

Set to True to take control of the call so that audio can be played for the call.

Cancel pending operations

Set to True to cancel any operations about to be performed on the call object. For example, if a prompt or audio file was about to be played, the play operation would be canceled.

Associate call with virtual station

This Boolean is set to true if the call is associated with a virtual station. By default, this parameter is false.

Exit Paths

Success

This step takes the Success exit path if the state is changed.

Failure

This step takes the Failure exit path if the state cannot be changed. This can occur for several reasons. Failure can occur if the call disconnects, if the call is picked up by a user (and is no longer under the control of the handler), the call ID is no longer valid (if the call is deallocated), or system resource limitation.

Send ADSI String

This Telephony tool sends two strings to a station queue to be displayed on an ADSI (Analog Display Services Interface) capable telephone. While the labels for this tool indicate that you should send a telephone number and name to display, you can send any string. The string is only sent when the phone is not connected to another CIC audio source (such as receiving dial tone, listening to voicemail, connected to a call, etc.).

To clear the text displayed on the phone, call another instance of this tool with empty Calling Number and Calling Name strings.

The station to which the strings are sent must be configured with the ADSI option selected. See ADSI in the Interaction Administrator online help for more information on configuring a station to receive ADSI information.

Inputs

Station Queue Identifier

The station on which to display the strings.

Calling Number to be Displayed

The first string to be displayed on the ADSI telephone.

Calling Name to be Displayed

The second string to be displayed on the ADSI telephone.

Exit Paths

Next

This tool always takes the Next exit path.

Set Datetime Attribute

This Telephony tool sets the datetime value for a specified call attribute.

Note: This tool does not support setting a datetime variable to a date prior to 1970/01/06.

Inputs

Call Identifier

The unique identifier for the call.

Call Attribute Name

The attribute for which the datetime value is being set.

Call Attribute Datetime Value

The datetime value for the call attribute.

Exit Paths

Next

This tool always takes the Next exit path.

Set Datetime Attributes

This Telephony tool sets the datetime values for multiple attributes of a call.

Note: This tool does not support setting a datetime variable to a date prior to 1970/01/06.

Inputs

Call Identifier

The unique identifier for the call.

Call Attribute Names

The list of attributes for which the datetime value is being set.

Call Attribute Datetime Values

The datetime values for the listed call attributes.

Exit Paths

Next

This tool always takes the Next exit path.

Set DTMF Password

This Telephony tool sets the password for a user. This password can be, but does not have to be, composed of DTMF digits.

Inputs

User Identifier

The user identifier for the user setting his or her password. The following examples demonstrate acceptable syntax:

```
"JohnD"  
"User:JohnD"  
"User Queue:JohnD"
```

Password (DTMF Digits)

The digits to use as the new password. This password can be, but does not have to be, composed of DTMF digits.

User's Current Password

The password (if any) that is to be replaced by this tool.

Exit Paths

Success

This path is taken if the user password is successfully set.

Invalid User Name

This path is taken if the user name entered is invalid.

Password in Password History

This path is taken if the new password is still in the user's password history and is not yet available for re-use.

Min Password Age

This path is taken if the password being replaced has not yet been in use for the minimum specified number of days.

Password Too Short

This path is taken if the password specified does not meet the minimum number of required digits.

Password Note Unique Enough

This path is taken if the password entered does not contain the minimum number of unique DTMF digits.

All Sequential Digits

This path is taken if the password is composed entirely of sequential DTMF digits, either ascending or descending.

Old Password Incorrect

This path is taken if the old password entered is incorrect.

Account Locked Out

This path is taken if the agent is locked out due to exceeding the number of allowable failed logon attempts.

Failure

This path is taken if the tool is unable to put the password into Directory Services.

Set State String

This Telephony tool step changes the state string of a call.

See [States](#) for information on state strings.

Inputs

Call Identifier

The unique identifier for the call.

New Call State

The state to which the call should be set.

Exit Paths

Success

This step takes the Success exit path if the state is changed.

Failure

This step takes the Failure exit path if the state cannot be changed. This can occur for several reasons. Failure can occur if the call disconnects, if the call is picked up by a user (and is no longer under the control of the handler), the call ID is no longer valid (if the call is deallocated), or there are system resource limitations.

Set User Status

This Telephony tool assigns a status to a user. The most common use for this tool is to assign an ACD agent who is not answering his or her phone a status of "ACD - Agent not answering." The statuses that can be assigned are listed below.

Inputs

User Identifier

The ID of the user whose status is to be changed.

Status Message Name

The status to be assigned to the user. Types are:

- ACD - Agent not answering
- Available
- Available (but connected to another call)
- On vacation
- Out of town
- Gone home
- At lunch
- Do not disturb
- Follow up
- Out of the office
- At a training session
- In a meeting

Date and Time for Status

The date and time information that accompanies the status. This information can be played in an IVR handler to tell a caller when a user will be available.

Is date in StatusDateTime set?

Set this to true if you want to include date information.

Is time in StatusDateTime set?

Set this to true if you want to include time information.

Auto Reset After x Seconds (x < 1 means do not reset)?

The number of seconds after which the user's status will reset. If set to a number less than 1, the user's status will not reset automatically.

Forwarding Number

The forwarding number for the user. This has no effect if a status date or time is being set or if auto-reset is set.

Exit Paths

Success

If this step executes successfully, this step takes the Success exit path.

Failure

If this step does not execute successfully, this step takes the Failure exit path.

Set Visual Indicator

This Telephony tool toggles (on or off) the indicator on an ADSI (Analog Display Services Interface) telephone. If the telephone is connected to another CIC audio source (such as receiving dial tone, listening to voicemail, connected to a call, etc.), the indicator is set when the station becomes available. It also keeps counts of new, old, and urgent messages.

The station on which the indicator is toggled must be configured with the ADSI option selected. See ADSI in the Interaction Administrator online help Station Configuration topic for more information on configuring a station to receive ADSI information.

Inputs

Station Queue Identifier

The station on which the indicator is toggled.

Turn Visual Indicator on?

Set to True to turn on the indicator. Set to False to turn off the indicator.

New message count

The number of unread messages to be displayed.

Old message count

The number of read messages to be displayed.

Urgent New Message Count

The number of unread, high priority messages to be displayed.

Urgent Old Message Count

The number of read, high priority messages to be displayed.

Exit Paths

Next

This tool always takes the Next exit path.

Start TDD

This Telephony tool was deprecated in CIC 4.0.

Station Answer

This Telephony tool picks up a call in alerting or held state on a station queue. It does not pick up calls in the connected state. This tool was created so that if multiple stations go off hook at the same time for a group ring, one station will not steal a call from another. For example, in a group ring situation, if three phones go off hook around the same time, the call should go to the first station and not to the other two.

Inputs

Call Identifier

The identifier for the call that is answered.

Station Queue Identifier

The identifier for the station queue on which the call resides.

Exit Paths

Success

This step takes the Success exit path if the operation is successful.

Failure

This step takes the Failure exit path if the station queue or call ID is not valid.

Station Audio

This Telephony tool makes a connection to a station queue. This connection creates a call object, and any type of audio can be played to that queue once the connection is made. You can also pass the connection to a [Station Place Call](#) tool.

Station audio calls made with this tool are only valid for a limited set of operations: play and input, station place call, and blind transfer.

Inputs

Station Queue Identifier

The station with which a connection is created.

User Identifier

If you specify a User Queue ID, the station audio call will be placed on that user queue in addition to the specified station queue.

Hide Call (Do not show on queues)

Set to True if you do not want the call to appear on a station queue.

Status Text

The Status Text input specifies a string which, if not empty, is the initial status string for the call. For the Off Hook handler, you should set it to "Manual Dialing."

Interrupt Current Call?

This Boolean determines whether or not the new connection will interrupt a currently connected call. By default, this parameter is set to "false."

Initiated by Phone?

This Boolean indicates whether or not the call was initiated by a phone, since hold requests for phone-initiated calls and client-initiated calls are handled differently. This parameter is set to "false" by default and is only set to "true" when the tool is called from inside the Station Off Hook initiator.

Outputs

Call Identifier

The unique identifier for the call. This call is created when a connection is established with the station queue.

Exit Paths

Success

This step takes the Success exit path if the Station Identifier is valid.

Failure

This path is taken if the operation fails. Reasons for failure can include currently active call or station audio call, or an invalid user queue. Insufficient voice resources can also cause this tool to fail if it is occurring frequently. The event log should indicate if this has been happening.

Station Connection Confirmation

This Telephony tool checks the status of a specified connection call.

Inputs

Connection Call ID

The unique identifier of the call waiting on confirmation.

Status

The status of the connection call.

Exit Paths

Next

This tool always takes the Next exit path.

Station Group Pickup

This Telephony tool enables handlers to implement the group pickup feature via a star code. On Polycom phones, this feature is available through the phone's menu and is initiated with SIP signaling.

The feature allows a user to pick up the longest alerting call in the station group for the specified Station Queue.

Inputs

Station Queue Identifier

The unique identifier for the station queue.

Exit Paths

Success

This tool takes the Success exit path if the operation is successful.

Failure

This tool takes the Failure exit path if the operation cannot be performed.

Station Pickup

This Telephony tool picks up a call alerting on a station queue. Use Station Pickup when you want a call to be connected to a station, just as if a CIC client pickup button was pressed. This tool was created to connect calls on analog stations not running a CIC client.

Inputs

Call Identifier

The identifier for the call that is picked up.

Station Queue Identifier

The identifier for the station queue on which the call resides.

Exit Paths

Success

This step takes the Success exit path if the call is picked up.

Failure

This tool can take the Failure exit path for several reasons. Failure can occur if the call disconnects, if the call is picked up by an CIC client user (and is no longer under the control of the handler), the call ID is no longer valid (if the call is deallocated), or there is a system resource limitation.

Station Place Call

This Telephony tool places a call on behalf of a station device so that once the call connects, it is connected to a station.

This tool generates an "Outgoing Call Request" event. In the current handler set, this starts the System_InitiateCallRequest handler.

Inputs

Call Identifier

This must be a call ID generated by the [Station Audio](#) tool.

Telephone Number

The telephone number to display to a CIC client user.

Note: This is not the number that is dialed.

Line Groups

This parameter takes a list of line groups (as configured in Interaction Administrator), or leave this parameter empty to use any available line group. In the default shipping handlers, the DialPlanEX subroutines returns a list of line groups to be used. You can pass that list in this parameter.

Station Queue Identifier

If this string is empty, the call is placed from the system queue. If the string identifies a station ("Station:Gagle", "Station Queue:Gagle" or just "Gagle"), the call is placed from the named station and if there is a logged in user on that station, the user's queue.

User/Workgroup Queue Identifier

Identifies the user or workgroup from which this call originated. This parameter is useful for auto-dialers, such as the Interaction Dialer, when the ACD Server determines to which agent a connected call should be sent.

Exit Paths

Success

If the operation is successful, this tool takes the Success exit path.

Failure

This tool takes the Failure exit path if the call ID is invalid or if the station queue ID is invalid.

Synchronous Answer

This Telephony tool establishes a connection with an incoming call. It differs from the [Answer](#) tool in that the GetCall command issued to the TS server is done synchronously, allowing the call to be transferred to a different system if the first answering system lacks the resources to perform IVR.

Inputs

Call Identifier

The unique identifier for the incoming call.

Exit Paths

Success

This path is taken if the call is answered.

Failure

This path is taken if the operation fails. This will happen if the Call ID is invalid.

Disconnect

This path is taken if the call is disconnected before the operation is complete.

Resources Unavailable

This path is taken if the system does not have enough resources available to handle the call.

System Queue

This Telephony tool removes an interaction from any queue(s) that contain it, and it generates a Transfer to System Queue event. This tool can access calls in any non-disconnected state.

Inputs

Call Identifier

The unique identifier for a call to be sent to the System queue.

Exit Paths

Next

This step always takes the Next exit path.

Transfer

This Telephony tool allows you to perform a consult transfer. This step connects two calls. For example, you receive Call1. Call1 asks to be transferred to person 2. You call person 2, which creates Call2. You now have Call1 and Call2. The Transfer tool connects those two calls. For immediately transferring calls without first creating Call2, see the Blind Transfer tool.

Inputs

Call Identifier

The unique identifier for the first call.

Call Identifier

The unique identifier for the second call. This call will be connected to the first call.

Use Putback (if available)

This Boolean, set appropriately by default by CIC according to the configured line's support for putting a call back on the originating system, is used to determine whether or not to use the putback feature if it is available for that line. If putback is available, then setting this parameter to True will cause the transfer to be attempted using the Putback operation. If the Putback operation fails within TS Server, a conventional transfer will be attempted.

Note: We strongly recommend that you do not change the default value of this parameter unless you know exactly when and why you need to override the default system value. This value is normally passed through the Transfer Request Initiator, but it can be changed there as well.

Exit Paths

Success

This step takes the Success exit path if the calls are connected.

Failure

This tool can take the Failure exit path for several reasons. Failure can occur if the call disconnects, if the call is picked up by a user (and is no longer under the control of the handler), the call ID is no longer valid (if the call is deallocated), or there is a system resource limitation.

Unbind Station

This Telephony tool unprovisions the current phone by clearing the MAC address for the phone in Interaction Administrator and causing the phone to reboot. This tool might be used, for example, when an administrator needs to provision a phone for a new user and the previous user's configuration remains.

Inputs

Real Station Queue Identifier

The unique identifier for the station.

Exit Paths

Success

This tool takes the Success exit path if the operation is successful.

Failure

This tool takes the Failure exit path if the operation cannot be performed.

Unhold Call

This Telephony tool takes a call off hold.

Inputs

Call Identifier

The unique ID of the call to remove from hold.

Station Queue Identifier

The unique ID of the station queue where the call is on hold.

Exit Paths

Success

This tool takes the Success exit path if the operation is successful.

Failure

This tool takes the Failure exit path if the operation cannot be performed.

User Login List

This Telephony tool returns a list of stations into which a given user is logged in. This tool was created for whisper functionality so that handlers can determine which stations should receive a whisper tone.

Inputs

User Identifier

The User for whom you want to return the station list. The User Identifier can be written as "MikeG", "User:MikeG" or "User Queue:MikeG".

Outputs

List of stations which should be alerted

A list of strings containing the stations into which the user is logged in.

Exit Paths

Next

This tool always takes the Next exit path.

Validate DTMF Password

This Telephony tool validates passwords entered over the telephone for remote voice mail retrieval.

Inputs

User Identifier

The user ID for which the password will be looked up. The following examples demonstrate acceptable syntax:

```
"JohnD"  
"User:JohnD"  
"User Queue:JohnD"
```

Password (DTMF digits)

The password entered by the user.

Exit Paths

Success

This step takes the Success exit path if the password matched the user ID.

User in Nag Period

This path is taken if the password is set to expire within a few days and the user is being reminded to change it.

Invalid User Name

This path is taken if the user name entered is invalid.

User Must Change Password

This path is taken if the nag period has ended and the user must change their password.

Password Expired

This path is taken if the password entered is no longer valid.

Invalid Password

This path is taken if the new password is still in the user's password history and not yet available for re-use, the password specified does not meet the minimum number of required digits, the password is composed entirely of sequential DTMF digits, or the password does not contain the minimum number of unique DTMF digits.

Account Locked Out

This path is taken if the agent is locked out due to exceeding the number of allowable failed logon attempts.

Failure

This step takes the Failure exit path if the user ID could not be found or if the password supplied was incorrect.

Verify Conference ID

This Telephony tool checks a Conference ID to ensure it hasn't been deallocated.

Inputs

Conference ID

The conference ID to check.

Exit Paths

Success

This path is taken if the Conference ID is valid.

Invalid

This path is taken if the Conference ID is invalid.

Verify Interaction ID

This Telephony tool checks an Interaction ID to ensure that it hasn't been deactivated. Deactivation of an interaction takes place 60 seconds after deallocation.

Inputs

Interaction ID

The Interaction ID to check.

Exit Paths

Success

This path is taken if the Interaction ID is valid.

Invalid

This path is taken if the Interaction ID is invalid.

Wait For Call On Queue

This Telephony tool waits for an interaction on a specified queue or queues.

Inputs

Call Identifier

The unique identifier for an interaction.

Timeout (seconds)

The number of seconds the tool will wait for the interaction.

List of Queue Ids

The unique identifiers for the queue or queues the tool will monitor while waiting for the interaction.

Exit Paths

Success

This path is taken if the specified interaction appears in one of the designated queues.

Failure

This path is taken if the operation fails.

Timeout

This path is taken if the timeout period ends before the specified interaction is located.

Wait For Disconnect

This Telephony tool causes the handler to pause until the specified call disconnects.

Inputs

Call Identifier

The unique identifier for the call.

Timeout

The number of seconds the tool will wait for the specified call to disconnect. A value of -1 gives this tool an infinite timeout period. A value of zero is also valid and will cause the tool to return immediately. A value of zero would be used if you don't need the handler to wait, but simply want to see if the call is in a disconnected state.

Exit Paths

Success

This path is taken if the specified call disconnects within the specified time period.

Timeout

This path is taken if the specified call does not disconnect within the specified time period.

Failure

This path is taken if the operation fails.

Wait For Monitor End

This Telephony tool pauses a handler until monitoring on the specified call ends or until the specified timeout period elapses.

Inputs

Call Identifier

The unique identifier for a monitor interaction.

Timeout (seconds)

The number of seconds the tool will wait for the monitor end.

Exit Paths

Monitor Disconnect

This path is taken if the party that initiated monitoring disconnects.

Target Disconnect

This path is taken if the monitored call disconnects.

Target Private

This path is taken if the call has been marked private by clicking the "Private" button in a CIC client.

Target Station Change

This path is taken if the call changes stations.

Stop Monitoring

This path is taken if call monitoring ends before the timeout period elapses.

Unknown

This path is taken if the tool exits for reasons other than call disconnection or monitor ending. Possible reasons for taking this path include if the call had already disconnected before this tool began, or if an incorrect Call Identifier was used.

Timeout

This path is taken if the timeout period elapses and the call is still being monitored.

Wait Wrap Up

This Telephony tool was previously used to pause the handler until a wrap-up code can be retrieved.

Note: This tool is no longer usable following IC 3.0 SU13.

Inputs

Call Identifier

The identifier of the call being waited on.

User Queue Identifier

The name of the user queue which this call is on

Exit Paths

Needed

This path is taken if the call has a wrap-up code that has not yet been entered by the CIC client.

Not Needed

This path is taken when a wrap-up code has been entered by the CIC client, if the call was removed from the user queue, or if the call has gone to the voicemail state.

Failure

This path is taken if the call does not exist, if the user queue does not exist, or if the call is not on the user queue.

Wink

This Telephony tool is used to send a wink on a digital line to a CO (Central Office) while the call is in the "Connected" state. You must, for this reason, define the wink as an "inverse wink" (at least one bit is high, low and high again). A wink is defined (for all T1/E1 spans, unless override in the DCM (Dialogic Configuration Manager)) in the spandti.prm file located in the \Dialogic\data directory; you must edit this file in order to change the definition of the wink. If you want to change the definition of the wink for a single span, you must edit the file and rename it for the span. The name of the new file must then be entered for the particular span into the parameter field in the misc table of the DCM. Only one wink definition can be defined for each E1/T1 line.

Any line on a T1/E1 span on which you change the wink definition (to be an inverse wink) cannot not be configured as "Wink start" since "Wink Start" uses a regular wink.

Inputs

Call Identifier

The identifier for the call on which the wink is sent.

Exit Paths

Next

This tool always takes the Next exit path.

Zone Page

This Telephony tool initiates a zone page. By dialing a short digit sequence (e.g., *88<zone>), a user, the page initiator, can begin a live one-way broadcast to stations in a zone. This zone can be made up of any combination of user queues, station queues, workgroup queues, and station groups.

At sites with poor network performance, there may be some significant delay between stations. In an environment where the listener can hear more than one phone at a time, this may result in a noticeable echo between the broadcasting phones.

A zone page terminates when one of the following occurs:

- the initiating station goes "on hook."
- the initiating station enters the termination digit.
- the timeout expires.

Notes: The Zone Page operation uses one IP resource per target phone on the CIC Server.

The Zone Page feature is only supported on Polycom SIP phones that support auto-answer (and are configured for auto-answer). See the list of supported IP phones at: <http://testlab.genesys.com/ProductsPage.aspx?ProductType=5>

Inputs

Call Identifier

The identifier for the call on which the zone page is sent. The call must be in a "connected" state.

List of Queue Identifiers

A list of fully scoped queue ID's to page. This list can be any combination of user queues, station queues, workgroup queues and station groups.

Page Origin

The remote name displayed on paged phones, if specified. If an empty string is passed, the station name configured in IA will be displayed.

Termination Keys

The telephone keys that terminate a zone page call. Terminating a zone page call in this way allows the initiating station to complete the page without going "on hook".

Timeout (seconds)

The number of seconds before an alerting zone page times out. If the zone page times out, the initiator and all paged stations are disconnected. By default this is set to -15.0.

Frequency 1 in Hertz (0:disable)

The frequency of the tone played to the page initiator. By default this is set to 350.

Frequency 1 Amplitude in dB (-40 thru 0)

The amplitude of the tone played to the page initiator. This is a number between -40 and 0 in dB. By default this is set to -10.

Tone Duration (0:continuous)

The duration in seconds of the tone played to the page initiator. To convert milliseconds to seconds, divide your millisecond value by 1000. For example, if you want the tone to play for 500 milliseconds, you would type 0.5 in this parameter. By default this is set to 0.5.

Exit Paths

Success

This path is taken if the zone page is delivered.

No Phones Available

The tool takes the No Phones Available exit path if there are no stations available to take the call. A station must be "on hook" to receive a zone page.

Invalid Queue

The tool takes the Invalid Queue exit path if there is an invalid queue in the list of queue identifiers parameter

Timeout

This tool takes the Timeout exit path if the zone page alerts longer than the number of seconds specified in the Timeout parameter.

Failure

This tool can take the Failure exit path for several reasons. Failure can occur if the tone or prompt generation to the initiator fails, the call ID is no longer valid (if the call is deallocated), or there is a system resource limitation.

UMF

Overview of Universal Messaging Facility

The Universal Messaging Facility provides a number of tools that a handler author can use to construct arbitrarily complex messages and send them to other handlers published on the same server. The sending of a message triggers one or more initiators. The initiators have inputs that allow the author to specify which handlers will trigger based on data contained in each message.

The originating handler can create a message and add elements of various types. Then it can choose to send the message as an event (asynchronously) to which it expects no reply or as a request (synchronously). When a message is sent as a request the toolstep in the originating handler waits for a message to be returned before continuing.

The receiving handler uses the tools that get data elements from the message. The receiving handler can in turn use the message construction tools to create and send a message to yet another handler or to send a response back to the original handler if appropriate.

NOTE: This subsystem depends on some basic CIC components as well as on Microsoft's MFC libraries. These items must be present in executable form at runtime, as well as represented by .lib and .h files at compile time.

Click on one of the tools below for more information about that tool.

[UMF Create Message](#)

[UMF Get Integer](#)

[UMF Get String](#)

[UMF Put Integer](#)

[UMF Put String](#)

[UMF Send Event](#)

[UMF Send Request](#)

[UMF Send Response](#)

UMF Create Message

This UMF tool creates a message in the system and provides a handle to it as an output. The handle is used as input to all other tools that operate on the message.

Outputs

Message Handle

A valid handle to the newly created message.

Exit Paths

Success

This path is taken if the message is successfully created.

Failure

This path is taken if the message creating fails.

UMF Get Integer

This UMF tool reads a data element from the message specified by the supplied handle.

Inputs

Message Handle

A valid handle to a message

Outputs

Value

The value that was read from the message. This output will not contain a valid value unless the success exit path is taken.

Exit Paths

Success

This path is taken if the value is successfully read from the message.

Failure

This path is taken if the operation fails or if the element being read from the message is not the correct type.

UMF Get String

This UMF tool reads a data element from the message specified by the supplied handle.

Inputs

Message Handle

A valid handle to a message.

Outputs

Value

The value that was read from the message. This output will not contain a valid value unless the success exit path is taken.

Exit Paths

Success

This path is taken if the value is successfully read from the message.

Failure

This path is taken if the operation fails or if the element being read from the message is not the correct type.

UMF Put Integer

This UMF tool adds a data element to the message specified by the supplied handle.

Inputs

Message Handle

A valid handle to a message.

Value

The value to be added to the message.

Exit Paths

Success

This path is taken if the value is successfully added to the message.

Failure

This path is taken if the operation fails.

UMF Put String

This UMF tool adds a data element to the message specified by the supplied handle.

Inputs

Message Handle

A valid handle to a message.

Value

The value to be added to the message.

Exit Paths

Success

This path is taken if the value is successfully added to the message.

Failure

This path is taken if the operation fails.

UMF Send Event

This UMF tool sends the message specified by the supplied handle to another handler on the same server. Which handler gets triggered by the message is determined by the event and object ids that are specified on this tool. Handler execution continues as soon as the message has been sent. No response is expected.

Inputs

Message Handle

A valid handle to a message.

Object ID

The value to use for the object ID that will be included with the message notification to the receiving handler. To trigger a handler this value must match the value specified in the Object ID input on the UMF [Message Received](#) initiator.

Event ID

The value to use for the event ID that will be included with the message notification to the receiving handler. to trigger a handler this value must match the value specified in the Notification Event input on the UMF Message Received initiator.

Exit Paths

Success

This path is taken if the message is successfully sent. This does not mean it was received by the intended handler, just that it was sent by the Notifier.

Failure

This path is taken if the operation fails.

UMF Send Request

This tool sends the message specified by the supplied handle to another handler on the same server. Which handler gets triggered by the message is determined by the event and object IDs that are specified of this tool. Handler execution continues only after a response is received or the timeout period expires.

Inputs

Message Handle

A valid handle to a message.

Object ID

The value to use for the object ID that will be included with the message notification to the receiving handler. To trigger a handler this value must match the value specified in the Object ID input on the [UMF Message Received](#) initiator.

Event ID

The value to use for the event ID that will be included with the message notification to the receiving handler. To trigger a handler this value must match the value specified in the Notification Event input on the UMF Message Received initiator.

Timeout

The number of milliseconds to wait for a response to the message. A value of zero means the timeout period is infinite.

Outputs

Response Message Handle

A handle to the response message if one was received in time. This output will only contain a valid value if the success path is taken.

Exit Paths

Success

This path is taken if the message is successfully sent and a response is received within the timeout period.

Failure

This path is taken if the operation fails.

Timeout

This path is taken if the timeout period expires before a response message has been received.

UMF Send Response

This UMF tool sends a response message specified by the supplied handle back to the source of the original message. One of the inputs is set from a value that can only be obtained from the initiator. This value is used to route the response to the correct recipient.

Inputs

Message Handle

A valid handle to a message.

Response Correlation ID

The value that is used to route the response back to the sender of the original message. This value is assigned by the server and can only be obtained from the Response Correlation ID output on the UMF Message Received initiator.

Exit Paths

Success

This path is taken if the message is successfully sent. This does not mean it was received by the intended handler, just that it was sent by the Notifier.

Failure

This path is taken if the operation fails.

VoiceXML

VoiceXML Tools Overview

The Voice XML interpreter optional feature is integrated into the PureConnect platform. Handlers process incoming calls and decide if a call needs to go through a VoiceXML application. The handler then invokes a tool that transfers the call to the VoiceXML Interpreter, and a VoiceXML session is activated. The URL of the VoiceXML document is specified by the session activation tool, and specifies which dialog to start. The VoiceXML Interpreter attempts to load and parse the document, and if it is successful, the Interpreter takes ownership of the call.

Depending on the dialog specified in the VXML document, the Interpreter either queues TTS or plays an audio prompt to the caller. The Interpreter then listens for DTMF or speech input.

Activating a VoiceXML session

A speech session is activated by an CIC Handler using one of the following four tools:

- [VoiceXML Initiate](#) - This tool sends a request to the VoiceXML Interpreter to initiate a session with the specified interaction, and the initial VoiceXML document URL is given as an argument. This tool issues a synchronous request.
- [VoiceXML Async Initiate](#) - This tool sends a request to the VoiceXML Interpreter to initiate a session with the specified interaction, and the initial VoiceXML document URL is given as an argument. This tool issues an asynchronous request.
- [VoiceXML Initiate Document](#) - This tool sends a request to the VoiceXML Interpreter to initiate a session with the specified interaction, and the initial VoiceXML document is given as an argument. This tool is most useful to interpret dynamically generated VoiceXML documents or documents read from a database. This tool issues a synchronous request.
- [VoiceXML Async Initiate Document](#) - This tool sends a request to the VoiceXML Interpreter to initiate a session with the specified interaction, and the initial VoiceXML document is given as an argument. This tool is most useful to interpret dynamically generated VoiceXML documents or documents read from a database. This tool issues an asynchronous request.
- [VoiceXML Async Cancel](#) - This VoiceXML tool sends a request to the VoiceXML Interpreter to cancel play/input operations. This tool issues a synchronous request.

Asynchronous and Synchronous activations

In asynchronous activations, the tool returns as soon as the VoiceXML session initiates and data cannot be sent back to the handler. This activation frees the handler for another session.

In synchronous activations, the tool waits until the VoiceXML session completes and the handler continues processing, allowing data to be sent back to the handler. For example the handler initiates a VoiceXML application to collect account information from a caller. The handler waits for the VoiceXML application to complete before transferring the caller and the account information to an agent.

Additional Information

For more information about VoiceXML integration in CIC, see the *VoiceXML Installation and Configuration Guide* in the PureConnect Documentation Library.

VoiceXML Async Initiate

This VoiceXML tool sends a request to the VoiceXML Interpreter to initiate a session with the specified interaction, and the initial VoiceXML document (URL) is given as an argument. The URL may contain a query.

The request is **asynchronous**. That means the tool returns as soon as the VoiceXML session has been initiated (the initial document is loaded, the ownership of the interaction transferred to the VoiceXML session, and the interpreter is ready to start).

The VoiceXML interpreter “owns” the interaction when the tool returns through the ‘Success’ exit. The handler may not perform operations against the interaction unless it re-acquires the ownership (in which case the VoiceXML session will abort immediately).

If the tool exits through the ‘Invalid Interaction’ or ‘Error’ exit, the handler remains the owner of the interaction.

Inputs

Interaction

The unique interaction identifier (e.g., CallID) to send to the VoiceXML interpreter.

Document URI

A string URI of the initial VoiceXML document.

Queued Plays Processing

Note: This parameter is not currently enabled but it is defined for future use.

This optional parameter specifies how to process pending prompt plays. There are three options:

0	Default - Play as "fetchaudio." Let the current prompts play until the VoiceXML session has started. Then, abort the current prompt as soon as the VoiceXML session is ready to play its own prompts.
1	Play to Completion. - Play the current prompt until it finishes, even if the VoiceXML session is started and ready to play its own prompts.
2	Abort. - The VoiceXML interpreter aborts any pending prompts immediately when it gets the call ownership. The Interpreter uses any global "fetchaudio" settings during the fetch. If the "fetchaudio" property has no value (default), nothing is played (silence).

Argument Names

An optional string list of argument name-value pairs that are passed to the interpreter as arguments.

Argument Values

An optional string list of argument values (name/value pairs) that correspond to the Argument Names. These arguments will be available in VoiceXML through 'session.name' or just simply 'name.'

Force Interaction Ownership

This checkbox explicitly tells the VoiceXML interpreter to force the interaction ownership.

Unchecked	Default. The VoiceXML interpreter only acquires ownership if the handler currently owns the interaction.
Checked	The VoiceXML interpreter always acquires the interaction ownership.

Outputs

Event Name

This string is the VoiceXML event that caused the interpreter to exit, or the reason for a session initiation failure.

It is an empty string if the session terminated through the <exit> element.

Event Message

This string is additional message text associated with the VoiceXML event ('_message' property).

Exit Paths

Success

The VoiceXML session completed successfully and the return value and/or result data (XML node) are valid.

Disconnected

The interaction disconnected during the session.

Lost Ownership

Some other entity has taken away the ownership of the interaction during the VoiceXML session.

Invalid Interaction

The interaction type is not supported by the VoiceXML interpreter.

Failure

An error occurred. That usually means an “error” event was thrown but not caught in the VoiceXML document. The name of the event is returned in ‘Event Name.’ If the initial document fails to load, an error.badfetch event with potentially additional decorations, such as error.badfetch.http.404, will be returned.

The ‘Event Message’ parameter usually contains additional information about what went wrong.

VoiceXML Async Initiate Document

This VoiceXML tool sends a request to the VoiceXML Interpreter to initiate a session with the specified interaction, and the initial VoiceXML document is given as an argument. This tool is most useful to interpret (simple) dynamically generated VoiceXML documents or documents read from a database or other store (e.g., Directory Services).

The request is **asynchronous**. That means the tool returns as soon as the VoiceXML session has been initiated (the initial document is loaded, the ownership of the interaction transferred to the VoiceXML session, and the interpreter is ready to start).

The VoiceXML interpreter “owns” the interaction when the tool returns through the ‘Success’ exit. The handler may not perform operations against the interaction unless it re-acquires the ownership (in which case the VoiceXML session aborts immediately).

If the tool exits through the ‘Invalid Interaction’ or ‘Error’ exit, the handler remains the owner of the interaction.

Inputs

Interaction

The unique interaction identifier (e.g., CallID) to send to the VoiceXML interpreter.

Document

This is an XML DOM node of the VoiceXML document to interpret. The argument may either be the document node or the <vxml> element.

Queued Plays Processing

Note: This parameter is not currently enabled but is defined for future use.

This optional parameter specifies how to process pending prompt plays. There are three options:

0	Default. - Play as “fetchaudio” Let the current prompts play until the VoiceXML session has started. Then abort the current prompt as soon as the VoiceXML session is ready to play its own prompts.
1	Play to Completion. - Play the current prompt until it finishes, even if the VoiceXML session is started and ready to play its own prompts.
2	Abort. - The VoiceXML interpreter aborts any pending prompts immediately when it gets the call ownership. The Interpreter uses any global “fetchaudio” settings during the fetch. If the “fetchaudio” property has no value (default), nothing is played (silence).

Argument Names

An optional string list of argument name-value pairs that are passed to the interpreter as arguments.

Argument Values

An optional string list of argument values (name/value pairs) that correspond to the Argument Names. These arguments are available in VoiceXML through 'session.name' or just simply 'name.'

Force Interaction Ownership

This checkbox explicitly tells the VoiceXML interpreter to force the interaction ownership.

Unchecked	Default. The VoiceXML interpreter only acquires ownership if the handler currently owns the interaction.
Checked	The VoiceXML interpreter always acquires the interaction ownership.

Outputs

Event Name

This string is the VoiceXML event that caused the interpreter to exit, or the reason for a session initiation failure.

It is an empty string if the session terminated through the `<exit>` element.

Event Message

This string is additional message text associated with the VoiceXML event ('_message' property).

Exit Paths

Success

The VoiceXML session completed successfully and the return value and/or result data (XML node) are valid.

Disconnected

The interaction disconnected during the session.

Lost Ownership

Some other entity has taken away the ownership of the interaction during the VoiceXML session.

Invalid Interaction

The interaction type is not supported by the VoiceXML interpreter.

Failure

An error occurred. That usually means an "error" event was thrown but not caught in the VoiceXML document. The name of the event is returned in 'Event Name.' If the initial document fails to load, an `error.badfetch` event with potentially additional decorations, such as `error.badfetch.http.404`, is returned.

The 'Event Message' parameter usually contains additional information about what went wrong.

VOICEXML INTERACT

This VoiceXML tool sends a request to the VoiceXML Interpreter to initiate a session with the specified interaction, and the initial VoiceXML document URL is given as an argument. This tool issues a **synchronous** request, which means the tool will block other events until the VoiceXML session completes.

If the tool returns through any exit other than 'Disconnected', 'Transferred', or 'Lost Ownership', the handler will (again) be the owner of the interaction.

Inputs

Interaction

The unique interaction identifier (e.g., CallID) to send to the VoiceXML interpreter.

Document URI

A string URI of the initial VoiceXML document.

Queued Plays Processing

Note: This parameter is not currently enabled but it is defined for future use.

This optional parameter specifies how to process pending prompt plays. There are three options:

0	Default - Play as "fetchaudio" Let the current prompts play until the VoiceXML session has started. Then abort the current prompt as soon as the VoiceXML session is ready to play its own prompts.
1	Play to Completion - Play the current prompt until it finishes, even if the VoiceXML session is started and ready to play its own prompts.
2	Abort - The VoiceXML interpreter will abort any pending prompts immediately when it gets the call ownership. The Interpreter will use any global "fetchaudio" settings during the fetch. If the "fetchaudio" property has no value (default), nothing is played (silence).

Argument Names

An optional string list of argument name-value pairs that are passed to the interpreter as arguments.

Argument Values

An optional string list of argument values (name/value pairs) that correspond to the Argument Names. These arguments will be available in VoiceXML through 'session.name' or just simply 'name'.

Force Interaction Ownership

This checkbox explicitly tells the VoiceXML interpreter to force the interaction ownership.

Unchecked	Default. The VoiceXML interpreter only acquires ownership if the handler currently owns the interaction.
Checked	The VoiceXML interpreter always acquires the interaction ownership.

Outputs

Return Value

This string is the result of the 'expr' expression of the <exit> element. Empty string if no return value.

Result Data

This XMLNode is an element of the result data of the session. (“namelist” argument of the <exit> element). It is a NULL node if there is no result data.

Event Name

This string is the VoiceXML event that caused the interpreter to exit, or the reason for a session initiation failure. It is an empty string if the session terminated through the <exit> element.

Event Message

This string is additional message text associated with the VoiceXML event (‘_message’ property).

Exit Paths

Success

The VoiceXML session completed successfully and the return value and/or result data (XML node) are valid.

Disconnected

The interaction disconnected during the session.

Transferred

The interaction was transferred to another destination during the session through the <transfer> element.

Lost Ownership

Some other entity has taken away the ownership of the interaction during the VoiceXML session.

Invalid Interaction

The interaction type is not supported by the VoiceXML interpreter.

Unhandled Event

This exit path is taken if the session terminated because some event other than “error” is thrown during the session and not handled by the VoiceXML application.

The ‘Event Name’ output parameter contains the name of the event and ‘Event Message’ may contain additional information.

Failure

An error occurred. That usually means an “error” event was thrown but not caught in the VoiceXML document. The name of the event is returned in ‘Event Name.’ If the initial document fails to load, an error.badfetch event with potentially additional decorations, such as error.badfetch.http.404, is returned.

The ‘Event Message’ parameter usually contains additional information about what went wrong.

VoiceXML Initiate Document

This VoiceXML tool sends a request to the VoiceXML Interpreter to initiate a session with the specified interaction, and the initial VoiceXML document is given as an argument. This tool is most useful to interpret dynamically generated VoiceXML documents or

documents read from a database. This tool issues a *synchronous* request, which means the tool will block other events until the VoiceXML session completes.

If the tool returns through any exit other than Disconnected, Transferred, or Lost Ownership, the handler will (again) be the owner of the interaction.

Inputs

Interaction

The unique interaction identifier (e.g., CallID) to send to the VoiceXML interpreter.

Document

This is an XML DOM node of the VoiceXML document to interpret. The argument may either be the document node or the <vxml> element.

Queued Plays Processing

Note: This parameter is not currently enabled but it is defined for future use.

This optional parameter specifies how to process pending prompt plays. There are three options:

0	Default. - Play as "fetchaudio." Let the current prompts play until the VoiceXML session has started. Then abort the current prompt as soon as the VoiceXML session is ready to play its own prompts.
1	Play to Completion. - Play the current prompt until it finishes, even if the VoiceXML session is started and ready to play its own prompts.
2	Abort. - The VoiceXML interpreter aborts any pending prompts immediately when it gets the call ownership. The Interpreter uses any global "fetchaudio" settings during the fetch. If the "fetchaudio" property has no value (default), nothing is played (silence).

Argument Names

An optional string list of argument name-value pairs that are passed to the interpreter as arguments.

Argument Values

An optional string list of argument values (name/value pairs) that correspond to the Argument Names. These arguments are available in VoiceXML through 'session.name' or just simply 'name.'

Force Interaction Ownership

This checkbox explicitly tells the VoiceXML interpreter to force the interaction ownership.

Unchecked	Default. The VoiceXML interpreter only acquires ownership if the handler currently owns the interaction.
Checked	The VoiceXML interpreter always acquires the interaction ownership.

Outputs

Return Value

This string is the result of the 'expr' expression of the <exit> element. Empty string if no return value.

Result Data

This XMLNode is an element of the result data of the session. (“namelist” argument of the <exit> element). It is a NULL node if there is no result data.

Event Name

This string is the VoiceXML event that caused the interpreter to exit, or the reason for a session initiation failure. It is an empty string if the session terminated through the <exit> element.

Event Message

This string is additional message text associated with the VoiceXML event (‘_message’ property).

Exit Paths

Success

The VoiceXML session completed successfully and the return value and/or result data (XML node) are valid.

Disconnected

The interaction disconnected during the session.

Transferred

The interaction was transferred to another destination during the session through the <transfer> element.

Lost Ownership

Some other entity has taken away the ownership of the interaction during the VoiceXML session.

Invalid Interaction

The interaction type is not supported by the VoiceXML interpreter.

Unhandled Event

This exit path is taken if the session terminated because some event other than “error” is thrown during the session and not handled by the VoiceXML application.

The ‘Event Name’ output parameter contains the name of the event and ‘Event Message’ may contain additional information.

Failure

An error occurred. That usually means an “error” event was thrown but not caught in the VoiceXML document. The name of the event is returned in ‘Event Name.’ If the initial document is not a valid VoiceXML document, an “error.badfetch” event will be returned.

The ‘Event Message’ parameter usually contains additional information about what went wrong.

VoiceXML Async Cancel

This VoiceXML tool sends a request to the VoiceXML Interpreter to cancel play/input operations.

The request is **asynchronous**. That means the tool returns as soon as the VoiceXML cancel request has been initiated.

The VoiceXML interpreter “owns” the interaction when the tool returns through the ‘Success’ exit. The handler may not perform operations against the interaction unless it re-acquires the ownership (in which case the VoiceXML session will abort immediately).

If the tool exits through the ‘Invalid Interaction’ or ‘Error’ exit, the handler remains the owner of the interaction.

Inputs

Interaction

The unique interaction identifier (e.g., CallID) to send to the VoiceXML interpreter.

Outputs

Event Name

This string is the VoiceXML event that caused the interpreter to exit, or the reason for a session initiation failure.

It is an empty string if the session terminated through the `<exit>` element.

Event Message

This string is additional message text associated with the VoiceXML event (‘_message’ property).

Exit Paths

Success

The VoiceXML cancel request completed successfully.

Web Interaction Tools

Overview of Web Interaction Tools

Web interaction tools are for building handlers that interact with people over the Internet. These interactions can be through a web browser or through Internet chat. CIC has the ability to generate custom web pages and pop-up chat utilities for a person browsing your web site.

Click on one of the following tools for more information about that tool:

[Alert Interaction](#)

[Chat Goto URL](#)

[Chat Send File](#)

[Conference Interaction](#)

[Consult Transfer](#)

[Create Interaction](#)

[Disconnect Interaction](#)

[Hold Interaction](#)

[Mute Interaction](#)

[Park Interaction](#)

[Pickup Interaction](#)

[Receive Text](#)

[Receive Text Async](#)

[Record](#)

[Send Text](#)

[Send Voicemail](#)

[Snip Recording](#)

[Snooze Interaction](#)

[Transfer Interaction](#)

Alert Interaction

This Web Interaction tool notifies a station queue of a web interaction (chat, collaboration, etc.) that needs to be picked up. This tool alerts any station the interaction's recipient is logged into. If the interaction's recipient is a workgroup, user, or line queue, any recipient monitoring that queue from a CIC client is alerted.

Alert Interaction times out if the interaction is not answered within a specific time period.

Alert Interaction changes the **state** of an interaction to Alerting. Only interactions in a state of Offering, On Hold, or Voice Mail can be acted upon by the Alert Interaction tool.

This tool also cancels any pending operations when transferring an ACD call to a user's queue.

Inputs

Interaction Identifier

The unique identifier of the interaction.

Timeout

The number of seconds this tool will wait before exiting via the Timeout exit path.

Is ACD Interaction

Set to True if the interaction is an ACD interaction.

ACD User

The user ID of the agent to receive this interaction if this is an ACD interaction.

Auto Answer

Set this value to True to have the interaction placed on the queue in a state of "Connected" if the station is off-hook. This is useful for agents who are not running a CIC client.

Outputs

Keys

The escape digit(s) pressed by the sender if the sender tries to cancel while the call is alerting.

Exit Paths

Success

This path is taken if the interaction is placed in a state of Alerting.

Failure

This path is taken if the operation fails.

Timeout

This path is taken if the designated timeout period expires before the interaction is picked up.

Chat Goto URL

This Web Interaction Tool sends a URL to participants of a web chat.

Inputs

Interaction Identifier

The unique identifier of the web interaction.

URL

The URL to display.

Exit Paths

Success

This path is taken if the URL is sent to all participants of the chat.

Failure

This path is taken if the operation fails.

Chat Send File

This Web Interaction tool sends text to a file.

Inputs

Interaction Identifier

The unique identifier of the web interaction.

Text

The text to be sent to the file.

Exit Paths

Success

This path is taken if the text is successfully sent.

Failure

This path is taken if the operation fails.

Conference Interaction

This Web Interaction tool creates a conference of two interactions, or adds an interaction to an existing conference.

Inputs

Target interaction to which to add a party

The unique ID of the interaction that will receive the new interaction.

Source interaction which to add to conference

The unique ID of the interaction being added to the conference.

Exit Paths

Success

This path is taken if the conference is successfully made.

Failure

This path is taken if the operation fails.

Consult Transfer

This Web Interaction tool performs a consult transfer of two web interactions. As with any transfer, you must define a source and a target.

Inputs

Target interaction to which to add a party

This is the interaction ID of the target of the consult transfer.

Source interaction which to add to conference

This is the interaction ID of the source of the consult transfer.

Exit Paths

Success

The consult transfer was successfully completed.

Failure

The consult transfer failed.

Create Interaction

This Web Interaction tool creates a chat or callback interaction from the system to an internal user.

Inputs

Interaction Type

The type of interaction being created. Valid options are "chat" or "callback."

Address Queue Identifier

The fully qualified name of the queue that the interaction will be placed on. For example, "User Queue:markm" or "Workgroup Queue:Marketing."

List of Attribute Names to Set

A list of strings containing the names of attributes to be set for this interaction.

List of Attribute Values to Set

A list of strings containing the values of the attributes to be set for this interaction. This list is parallel to the list of names. For example, if the "List of Attribute Names to Set" parameter contains three attribute names, then this parameter should contain three values; the first value will be assigned to the first name, the second value to the second name, and the third value to the third name.

Outputs

Interaction Id

The unique identifier for the newly created web interaction.

Exit Paths

Success

This path is taken if the interaction is successfully created.

Failure

This path is taken if the operation fails.

Disconnect Interaction

This Web Interaction tool is used to disconnect any Web interaction.

Inputs

Interaction Identifier

The unique identifier of the web interaction.

Exit Paths

Success

This path is taken if the interaction is successfully disconnected.

Failure

This path is taken if the operation fails.

Hold Interaction

This Web Interaction tool places a web interaction on hold. Interactions that are placed on hold must be in a connected state.

Inputs

Interaction Identifier

The unique identifier of the web interaction.

Exit Paths

Success

This path is taken if the interaction is successfully placed on hold.

Failure

This path is taken if the interaction fails.

Mute Interaction

This Web Interaction tool turns off any sound input from the user's end of a web interaction. This allows the user to screen unwanted sounds or conversations from without having to put the interaction on hold. This same tool is also used to reactivate sound input for muted interactions.

Inputs

Interaction Identifier

The unique identifier of the web interaction.

Mute Action (On/Off)

Set to True to turn muting on. Set to False to turn muting off.

Exit Paths

Success

This path is taken if the interaction is successfully muted or unmuted.

Failure

This path is taken if the operation fails.

Park interaction

This Web Interaction tool parks a web interaction (chats, callbacks, or collaborations) on a user queue you specify. The Park Interaction tool allows a handler to re-park an interaction after the original park operation has timed out. This tool allows handlers to differentiate between parked interactions timing out and held interactions timing out. For example, when a parked chat timeout occurs, the Held Call Timer initiator is triggered (just as it is for held calls). If the call was parked, the ParkedFlag output parameter in the Initiator is set to true. The handler could then give the caller the option to re-park.

Inputs

Interaction Identifier

The unique identifier for the web interaction.

Address Queue Identifier

The queue on which the interaction is to be parked.

Exit Paths

Success

This path is taken if the call is successfully parked on the designated queue.

Failure

This path is taken if the operation fails.

Pickup Interaction

This Web Interaction tool picks up a web interaction (e.g., chat, callback, or collaboration) that is on hold. Interactions that are not in a state of On Hold cannot be picked up. This step does not pick up an interaction that has already been picked up by a user.

Inputs

Interaction Identifier

The unique identifier for the web interaction.

Exit Paths

Success

This path is taken if the web interaction is successfully picked up.

Failure

This path is taken if the operation fails.

Receive Text

This Web Interaction tool receives text from a remote web interaction participant.

Inputs

Interaction Identifier

The unique identifier for the web interaction.

Timeout

The number of seconds the tool will wait to receive the text before taking the Timeout exit path.

Outputs

Text

The text received from the remote web interaction participant.

Exit Paths

Success

This path is taken if the text is successfully received.

Failure

This path is taken if the operation fails.

Timeout

This path is taken if the specified Timeout period expires before any text is received.

Receive Text Async

This Web Interaction tool receives text from a remote web interaction participant.

Inputs

Interaction Identifier

The unique identifier for the web interaction.

Timeout

The number of seconds the tool waits to receive the text before calling the Complete Async Receive Text initiator.

Context Value

The tool sends this value to the Complete Async Receive Text from an Interaction initiator to identify the source of the Receive Text request. For example, the value can be the name of a handler.

Escape Only

Set this optional value to TRUE to return only the escape text. This value is ignored when the Escape Text field is not specified, and therefore any string is treated as an escape string. The default value for the Escape Only input is FALSE.

Escape Text

Optional. When this text is matched, the Receive Text operation stops. You can use this input to specify an escape string to leave a message and return the escaped return code.

Exit Paths

Success

This path is taken if the text is successfully received.

Failure

This path is taken if the operation fails.

Note: Only one instance of this tool is allowed at a time. Calling an instance while another is already running results in the running instance being cancelled and failing to trigger the initiator.

Record

This Web Interaction tool starts and stops recording of a web interaction, just as if a CIC client user had pressed the Record button. If this tool stops recording, the recording object is disconnected and the standard OnDisconnect handlers fire to process the recording and send it to the appropriate party.

Inputs

Interaction Identifier

The unique identifier of the web interaction.

Recording Party

The user requesting the recording.

Supervisory Flag

Set to True if the requestor has supervisory status.

Record Action

Whether to Start or Stop recording.

Exit Paths

Success

This path is taken if the recording is successfully started or stopped.

Failure

This path is taken if the operation fails.

Send Text

This Web Interaction tool sends text to a remote web interaction participant.

Inputs

Interaction Identifier

The unique identifier of the web interaction.

Text

The text to be sent to the remote participant.

Exit Paths

Success

This path is taken if the text is successfully sent.

Failure

This path is taken if the operation fails.

Send To Voicemail

This Web Interaction tool sends a chat or other web interaction into chat mail.

Input

Interaction Identifier

The unique identifier of the web interaction.

Exit Paths

Success

This path is taken if the interaction is successfully sent to chat mail.

Failure

This path is taken if the operation fails.

Snip Recording

This Web Interaction tool starts and stops an Interaction Recorder snippet recording on an interaction.

Inputs

Interaction Identifier

The identifier of the interaction to start or stop recording.

User Id

The identifier for the user associated with the interaction.

User Name

The user name associated with the interaction.

User Extension

The user extension associated with the interaction.

Station Id

The station ID to start or stop recording.

Supervisory Flag

Set to True or False to indicate whether the requestor has supervisory status

Record Action

Whether to Start or Stop recording.

Exit Paths

Success

This path is taken if the interaction is successfully started or stopped.

Failure

This path is taken if the operation fails.

Snooze Interaction

This Web Interaction tool places a web interaction in a "Snoozed" state. For example, if attempts to reach the callback requestor are not successful, an agent can decide to retry the callback request later and place the callback request into a "Snoozed" state.

When an agent places a web interaction in a "Snooze" state, the Interaction Retry Later Attempt initiator starts the System_RetryInteractionLater handler to call the Snooze Interaction toolstep to place the interaction in the "Snooze" state. When the "Snooze" period ends, the Interaction Snooze Timed Out initiator starts the System_RetryInteractionTimeout handler to place the interaction in a queue to start ACD processing.

Inputs

Interaction Identifier

The unique identifier of the interaction.

Time

The datetime variable to indicate how long the interaction remains in a "Snoozed" state.

Exit Paths

Success

This path is taken if the tool successfully places the interaction in a "Snoozed" state.

Failure

This path is taken if the operation fails.

Transfer Interaction

This Web Interaction tool transfers a web interaction from one queue to another. Valid queue types include the User and Workgroup queues.

Inputs

Interaction Identifier

The unique identifier of the web interaction.

Address Queue Identifier

The queue to which the interaction is to be transferred.

Outputs

Outbound Identifier

This parameter is not populated by web interactions.

Exit Paths

Success

This path is taken if the web interaction is successfully transferred to the designated queue.

Failure

This path is taken if the operation fails.

WebSphere MQ

Overview of WebSphere MQ Tools

IBM's WebSphere MQ (formerly called MQSeries) message queue software enables business applications to exchange information across different operating system platforms. The PureConnect's MQ tools allow handlers to interact with a WebSphere MQ Client. If you are not already using WebSphere MQ applications at your company, then you will never need to use these tools.

The documentation for the CIC MQ tools offers brief explanations, but refers you to your WebSphere MQ documentation for details. This is because the CIC MQ tools are essentially a handler interface to the WebSphere MQ API.

These tools require that either the WebSphere MQ Client (recommended) or the WebSphere MQ server (i.e, QueueManager) be installed and running on the CIC server. If the MQ Client is installed on the CIC server, you must install the MQ server (QueueManager) on a separate server. If the MQ server is installed on the CIC server, the MQ Client can be installed on a separate server. You can also install both the MQ Client and server on the CIC server, but we do not recommend that, unless the CIC server is a very powerful server. The MQ server can run as a service, and you should see your WebSphere MQ documentation for more information on installing and running WebSphere MQ Client as a service.

IBM maintains online versions of all WebSphere MQ documentation on their product web site. Refer to the WebSphere MQ Client and Administrators Guide for more information on WebSphere MQ and WebSphere MQ server installation.

Click on one of the following CIC MQ tools for more information on that tool:

Note: MQSeries tools in Interaction Designer are deprecated and the IBM libraries are not updated anymore.

[MQ Begin](#)

[MQ Close](#)

[MQ Commit](#)

[MQ Connect](#)

[MQ Disconnect](#)

[MQ Extended Get](#)

[MQ Extended Put](#)

[MQ Flush](#)

[MQ Get](#)

[MQ Open](#)

[MQ Put](#)

[MQ Rollback](#)

MQ Begin

This WebSphere MQ tool executes the MQBegin call to begin a unit of work that is coordinated by the queue manager, and that may involve external resource managers. See the MQBegin call documentation in your *WebSphere MQ Application Programming Guide* for more information.

Inputs

Hconn

The connection handle to the Queue Manager. The [MQConnect](#) tool generates this connection handle. This value corresponds to the Hconn input parameter in the MQBegin call.

Outputs

Reason Code

For each call, a completion code and a reason code are returned by the queue manager or by an exit routine, to indicate the success or failure of the call. This value corresponds to the Reason code parameter in the MQBegin call. The *WebSphere MQ Application Programming Guide* contains a list of reason codes and their meanings.

Exit Paths

Success

This step takes the Success exit path if the completion code indicates success.

Failure

This step takes the Failure exit path if the completion code indicates failure. The reason for tool failure is contained in the Reason Code parameter.

MQ Close

This WebSphere MQ tool executes the MQClose call to relinquish access to an object, and is the opposite of the MQ Open tool. See the MQClose call documentation in your WebSphere MQ Application Programming Guide for more information.

Inputs

Hconn

The connection handle to the Queue Manager. The [MQConnect](#) tool generates this connection handle. This value corresponds to the Hconn input parameter in the MQClose call.

Hobj

The handle to the object that is being closed. The object can be of any type. This value corresponds to Hobj in the MQClose call. The [MQ Open](#) tool generates Hobj values. On successful completion, the queue manager sets this parameter to a value that is not a valid handle for the environment.

Options

Options that control how the object is closed. You may specify only one option. Valid options are listed in the *MQSeries Application Programming Reference* guide. This value corresponds to the Options parameter in the MQClose call.

Outputs

Reason Code

For each call, a completion code and a reason code are returned by the queue manager or by an exit routine, to indicate the success or failure of the call. This value corresponds to the Reason code parameter in the MQClose call. The WebSphere MQ Application Programming Guide contains a list of reason codes and their meanings.

Exit Paths

Success

This step takes the Success exit path if the completion code indicates success.

Failure

This step takes the Failure exit path if the completion code indicates failure. The reason for tool failure is contained in the Reason Code parameter.

MQ Commit

This WebSphere MQ tool executes the MQCMIT call to tell the queue manager that the application has reached a syncpoint, and that all of the message gets and puts that have occurred since the last syncpoint are to be made permanent. Messages put as part of a unit of work are made available to other applications; messages retrieved as part of a unit of work are deleted. See the MQCMIT call documentation in your WebSphere MQ Application Programming Guide for more information.

Inputs

Hconn

The connection handle to the Queue Manager. The [MQConnect](#) tool generates this connection handle. This value corresponds to the Hconn input parameter in the MQCMIT call.

Outputs

Reason Code

For each call, a completion code and a reason code are returned by the queue manager or by an exit routine, to indicate the success or failure of the call. This value corresponds to the Reason code parameter in the MQCMIT call. The WebSphere MQ Application Programming Guide contains a list of reason codes and their meanings.

Exit Paths

Success

This step takes the Success exit path if the completion code indicates success.

Failure

This step takes the Failure exit path if the completion code indicates failure. The reason for tool failure is contained in the Reason Code parameter.

MQ Connect

This WebSphere MQ tool executes the MQCONN call to connect an application program to a queue manager. MQ Connect provides a queue manager connection handle (Hconn), which is used by other MQSeries tools. See the MQCONN call documentation in your WebSphere MQ Application Programming Guide for more information.

Inputs

QMgrName

Name of queue manager. The name specified must be the name of a *connectable* queue manager. See the QMgrName parameter documentation in the *MQSeries Application Programming Reference* guide for syntax guidelines.

ConnectOptions (MQCNO)

An integer input parameter used with the MQCNO structure. This parameter is specific to the Options field during an MQCONNX connect operation, and defaults to 64.

If MQCNO_NONE is specified as input, the tool uses the MQCONN call to connect an application to a queue manager. Otherwise, it uses the MQCONNX call.

The following table lists the connect option fields in MQCNO:

Field	Description	Topic
StrucId	Structure identifier	StrucId
Version	Structure version number	Version
Options	Options that control the action of MQCONN	Options

The Options field in an MQCNO structure is of type MQLONG and corresponds to an integer in Interaction Designer. For more information about MQCNO options, see the WebSphere MQ documentation on IBM's Knowledge Center at:

https://www.ibm.com/support/knowledgecenter/SSFKSJ_7.0.1/com.ibm.mq.helphome.v70.doc/WelcomePagev7r0.htm

Use connection cache

Select this check box to use connection caching.

Outputs

Hconn

The connection handle that can be used by other MQSeries tools. This value corresponds to the Hconn parameter in the MQCONN call.

Reason Code

For each call, a completion code and a reason code are returned by the queue manager or by an exit routine, to indicate the success or failure of the call. This value corresponds to the Reason code parameter in the MQCONN call. The WebSphere MQ Application Programming Guide contains a list of reason codes and their meanings.

Exit Paths

Success

This step takes the Success exit path if the completion code indicates success.

Failure

This step takes the Failure exit path if the completion code indicates failure. The reason for tool failure is contained in the Reason Code parameter. This step will also fail if the Queue Manager isn't running or is configured incorrectly, or if the name specified in the QMgrName is incorrect.

MQ Disconnect

This WebSphere MQ tool executes the MQDISC call to break the connection between the queue manager and the application program. See the MQDISC call documentation in your WebSphere MQ Application Programming Guide for more information.

Inputs

Hconn

The connection handle to the Queue Manager to disconnect. The [MQConnect](#) tool generates this connection handle. This value corresponds to the Hconn input parameter in the MQDISC call.

Use connection cache

Select this check box to use connection caching.

Outputs

Reason Code

For each call, a completion code and a reason code are returned by the queue manager or by an exit routine, to indicate the success or failure of the call. This value corresponds to the Reason code parameter in the MQDISC call. The WebSphere MQ Application Programming Guide contains a list of reason codes and their meanings.

Exit Paths

Success

This step takes the Success exit path if the completion code indicates success.

Failure

This step takes the Failure exit path if the completion code indicates failure. The reason for tool failure is contained in the Reason Code parameter.

MQ Extended Get

This WebSphere MQ tool executes the MQGET call to retrieve a message from a local queue that has been opened using the MQOPEN call. See the MQGET call documentation in your WebSphere MQ Application Programming Guide for more information.

The function of MQ Extended Get is identical to [MQ Get](#), except that MQ Extended Get returns all MQGET call output parameters while MQ Get returns only the most commonly used output parameters.

Inputs

Hconn

The connection handle to the Queue Manager. The [MQConnect](#) tool generates this connection handle. This value corresponds to the Hconn input parameter in the MQGET call.

Hobj

The object handle identifying the queue from which a message is retrieved. The Hobj is generated by the MQ Open tool. This value corresponds to the Hobj parameter in the MQGET call.

Options (MQGMO)

Options that control the action of MQGET. See MQGMO - Get-Message options in the *MQSeries Application Programming Reference* guide for a list of valid options. This value corresponds to the GetMesgOpts parameter in the MQGET call.

WaitInterval (MQGMO)

The approximate time, expressed in milliseconds, that the MQGET call waits for a suitable message to arrive..

Search MsgId

See the Message-Identifier options in the MQMD - Message Descriptor documentation for more information on setting this value.

Search CoreId

See the Correlation-Identifier options in the MQMD - Message Descriptor documentation for more information on setting this value.

Max. Message Length Estimate

The maximum message length in bytes. See MaxMsgLength in the Attributes for Local Queues and Model Queues for more information on setting this value.

Buffer handle

If your message to get is a binary buffer constructed via the Buffer Tools, pass in the buffer handle here. Any non-zero number in the Buffer Handle implies a binary buffer. Set the Buffer Handle to 0 for a text buffer.

Outputs

Note: The values of the following output parameters are defined in the MQMD - Message Descriptor portion of the WebSphere MQ Application Programming Guide.

Message

If your message to get is all textual, pass in an empty string here; it will be filled-in with the message contents on return. Note: The Buffer Tools also support textual buffers, but it is generally easier to use a string for this.

See the cross-reference note above.

Report

See note above.

MsgType

See note above.

Expiry

See note above.

Feedback

See note above.

Format

See note above.

Encoding

See note above.

CodedCharSetId

See note above.

Priority

See note above.

Persistence

See note above.

MsgId

See note above.

CoreId

See note above.

BackoutCount

See note above.

ReplyToQ

See note above.

ReplyToQMgr

See note above.

UserIdentifier

See note above.

AccountingToken

See note above.

AppIdentityData

See note above.

PutAppIType

See note above.

PutAppIName

See note above.

PutDate

See note above.

PutTime

See note above.

ApplOriginData

See note above.

ResolvedQName (MQGMO)

See note above.

Reason Code

For each call, a completion code and a reason code are returned by the queue manager or by an exit routine, to indicate the success or failure of the call. This value corresponds to the Reason code parameter in the MQGET call. The WebSphere MQ Application Programming Guide contains a list of reason codes and their meanings.

Exit Paths

Success

This step takes the Success exit path if the completion code indicates success.

Failure

This step takes the Failure exit path if the completion code indicates failure. The reason for tool failure is contained in the Reason Code parameter.

MQ Extended Put

This WebSphere MQ tool executes the MQPUT call to put a message on a queue or distribution list. The queue or distribution list must already be open. See the MQPUT call documentation in your WebSphere MQ Application Programming Guide for more information.

The function of MQ Extended Put is identical to [MQ Put](#), except that MQ Extended Put sends all MQPUT call input parameters while MQ Put sends only the most commonly used input parameters.

Inputs

Hconn

The connection handle to the Queue Manager. The [MQConnect](#) tool generates this connection handle. This value corresponds to the Hconn input parameter in the MQPUT call.

Hobj

The object handle identifying the queue from which a message is retrieved. The Hobj is generated by the MQ Open tool. This value corresponds to the Hobj parameter in the MQPUT call.

Note: The values of the following input parameters are defined in the MQPUT and MQMD sections of the WebSphere MQ Application Programming Guide.

Buffer Handle

If your message to put is a binary buffer constructed via the Buffer Tools, pass in the buffer handle here. Any non-zero number in the Buffer Handle implies a binary buffer. Set the Buffer Handle to 0 for a text buffer.

Message

If your message to put is all textual, pass in a string containing the buffer contents here. Note: The Buffer Tools also support constructing a textual buffer, but it is generally easier to use a string for this. See the cross-reference Note above.

Options (MQPMO)

See note above.

Content (MQPMO)

See note above.

Report

See note above.

MsgType

See note above.

Expiry

See note above.

Feedback

See note above.

Format

See note above.

Encoding

See note above.

CodedCharSetId

See note above.

Priority

See note above.

Persistence

See note above.

MsgId

See note above.

CoreId

See note above.

ReplyToQ

See note above.

ReplyToQMgr

See note above.

UserIdentifier

See note above.

AccountingToken

See note above.

AppIdentityData

See note above.

PutAppIType

See note above.

PutAppIName

See note above.

PutDate

See note above.

PutTime

See note above.

AppIOriginData

See note above.

Outputs

MsgId

See note above.

CoreId

See note above.

ResolvedQName (MQPMO)

See note above.

ResolvedQMgrName (MQPMO)

See note above.

Reason Code

For each call, a completion code and a reason code are returned by the queue manager or by an exit routine, to indicate the success or failure of the call. This value corresponds to the Reason code parameter in the MQPUT call. The WebSphere MQ Application Programming Guide contains a list of reason codes and their meanings.

Exit Paths

Success

This step takes the Success exit path if the completion code indicates success.

Failure

This step takes the Failure exit path if the completion code indicates failure. The reason for tool failure is contained in the Reason Code parameter.

MQ Flush

This tool flushes the MQ message queue.

Outputs Tab

Reason Code

For each call, a completion code and a reason code are returned by the queue manager or by an exit routine, to indicate the success or failure of the call. The *WebSphere MQ Application Programming Guide* contains a list of reason codes and their meanings.

Exit Paths

Success

This path is taken if the message queue is successfully flushed.

Failure

This path is taken if the message queue was not successfully flushed.

MQ Get

This WebSphere MQ tool executes the MQGET call to retrieve a message from a local queue that has been opened using the MQOPEN call. See the MQGET call documentation in your WebSphere MQ Application Programming Guide for more information.

The function of [MQ Extended Get](#) is identical to MQ Get, except that MQ Extended Get returns all MQGET call output parameters while MQ Get returns only the most commonly used output parameters.

Inputs

Hconn

The connection handle to the Queue Manager. The [MQConnect](#) tool generates this connection handle. This value corresponds to the Hconn input parameter in the MQGET call.

Hobj

The object handle identifying the queue from which a message is retrieved. The Hobj is generated by the MQ Open tool. This value corresponds to the Hobj parameter in the MQGET call.

Options (MQGMO)

Options that control the action of MQGET. See MQGMO - Get-Message options in the *MQSeries Application Programming Reference* guide for a list of valid options. This value corresponds to the GetMesgOpts parameter in the MQGET call.

WaitInterval (MQGMO)

The approximate time, expressed in milliseconds, that the MQGET call waits for a suitable message to arrive..

Search CoreId

See the Correlation-Identifier options in the MQMD - Message Descriptor documentation for more information on setting this value.

Outputs

Note: The values of the following output parameters are defined in the MQMD - Message Descriptor portion of the *MQSeries Application Programming Reference* guide (<http://www.software.ibm.com/ts/mqseries/library/manuals/>).

Message

See note above.

MsgType

See note above.

Feedback

See note above.

MsgId

See note above.

CoreId

See note above.

ReplyToQ

See note above.

ReplyToQMgr

See note above.

Reason Code

For each call, a completion code and a reason code are returned by the queue manager or by an exit routine, to indicate the success or failure of the call. This value corresponds to the Reason code parameter in the MQGET call. The WebSphere MQ Application Programming Guide contains a list of reason codes and their meanings.

Exit Paths

Success

This step takes the Success exit path if the completion code indicates success.

Failure

This step takes the Failure exit path if the completion code indicates failure. The reason for tool failure is contained in the Reason Code parameter.

MQ Open

This WebSphere MQ tool executes the MQOPEN call to establish access to a queue or queue manager. See the MQOPEN call documentation in your WebSphere MQ Application Programming Guide for more information.

Inputs

Hconn

The connection handle to the Queue Manager. The [MQConnect](#) tool generates this connection handle. This value corresponds to the Hconn input parameter in the MQOPEN call.

ObjectType (MQOD)

The structure that identifies the object to be opened. See MQOD - Object Descriptor documentation in the WebSphere MQ Application Programming Guide for more information.

ObjectName (MQOD)

See MQOD - Object Descriptor documentation in the WebSphere MQ Application Programming Guide for more information.

ObjectQMgrName (MQOD)

See MQOD - Object Descriptor documentation in the WebSphere MQ Application Programming Guide for more information.

DynamicQName (MQOD)

See MQOD - Object Descriptor documentation in the WebSphere MQ Application Programming Guide for more information.

Options

See MQOPEN documentation in the WebSphere MQ Application Programming Guide for more information.

Outputs

Hobj

See MQOPEN documentation in the WebSphere MQ Application Programming Guide for more information.

See MQOD - Object Descriptor documentation in the WebSphere MQ Application Programming Guide for more information.

ObjectName (MQOD)

See MQOD - Object Descriptor documentation in the WebSphere MQ Application Programming Guide for more information.

Reason Code

For each call, a completion code and a reason code are returned by the queue manager or by an exit routine, to indicate the success or failure of the call. This value corresponds to the Reason code parameter in the MQOPEN call. The WebSphere MQ Application Programming Guide contains a list of reason codes and their meanings.

Exit Paths

Success

This step takes the Success exit path if the completion code indicates success.

Failure

This step takes the Failure exit path if the completion code indicates failure. The reason for tool failure is contained in the Reason Code parameter.

MQ Put

This WebSphere MQ tool executes the MQPUT call to put a message on a queue or distribution list. The queue or distribution list must already be open. See the MQPUT call documentation in your WebSphere MQ Application Programming Guide for more information.

The function of [MQ Extended Put](#) is identical to MQ Put, except that MQ Extended Put sends all MQPUT call input parameters while MQ Put sends only the most commonly used input parameters.

Inputs

Hconn

The connection handle to the Queue Manager. The [MQConnect](#) tool generates this connection handle. This value corresponds to the Hconn input parameter in the MQPUT call.

Hobj

The object handle identifying the queue from which a message is retrieved. The Hobj is generated by the MQ Open tool. This value corresponds to the Hobj parameter in the MQPUT call.

Note: The values of the following input parameters are defined in the MQPUT and MQMD sections of the WebSphere MQ Application Programming Guide.

Message

See note above.

Options (MQPMO)

See note above.

Report

See note above.

MsgType

See note above.

Expiry

See note above.

Feedback

See note above.

Priority

See note above.

CoreId

See note above.

ReplyToQ

See note above.

ReplyToQMgr

See note above.

Outputs

MsgId

See note above.

CoreId

See note above.

Reason Code

For each call, a completion code and a reason code are returned by the queue manager or by an exit routine, to indicate the success or failure of the call. This value corresponds to the Reason code parameter in the MQPUT call. The WebSphere MQ Application Programming Guide contains a list of reason codes and their meanings.

Exit Paths

Success

This step takes the Success exit path if the completion code indicates success.

Failure

This step takes the Failure exit path if the completion code indicates failure. The reason for tool failure is contained in the Reason Code parameter.

MQ Rollback

This WebSphere MQ tool executes the MQBACK call to tell queue manager that all of the message gets and puts that have occurred since the last syncpoint are to be backed out. Messages put as part of a unit of work are deleted; messages retrieved as part of a unit of work are reinstated on the queue. See the MQPUT call documentation in your WebSphere MQ Application Programming Guide for more information.

Inputs

Hconn

The connection handle to the Queue Manager. The [MQConnect](#) tool generates this connection handle. This value corresponds to the Hconn input parameter in the MQPUT call.

Outputs

Reason Code

For each call, a completion code and a reason code are returned by the queue manager or by an exit routine, to indicate the success or failure of the call. This value corresponds to the Reason code parameter in the MQBACK call. The WebSphere MQ Application Programming Guide contains a list of reason codes and their meanings.

Exit Paths

Success

This step takes the Success exit path if the completion code indicates success.

Failure

This step takes the Failure exit path if the completion code indicates failure. The reason for tool failure is contained in the Reason Code parameter.

XML

XML Add Schema

This XML tool adds a schema to the schema cache of the document. The schema cache can be used to associate a schema with a namespace. For example, you can load a document with 'Resolve Externals' and 'Validate On Parse' set to False. By adding pre-loaded schemas to the schema cache for schemas that require external resolving, you can avoid the potentially expensive step of resolving external references (for example, if the schema is located on a remote machine). After adding the schemas, you can invoke the [XML Validate Document](#) tool to validate the document against the schemas.

Noted: Schemas in the schema cache will have precedence over schemas already in the document. This therefore allows validating a document against a different schema than it was originally loaded with.

Adding a schema will automatically disable DTD processing, as you cannot use both in the same document.

The namespaces of Schemas added with this tool will only be included in the list returned by the [XML Get Namespaces](#) tool, if they are actually used (referenced) in the document.

Inputs

Document

Node of document to whose schema cache the schema is to be added.

Namespace URI

Namespace URI of the Schema to add.

Schema

Node of the Schema document to add. If not specified, schema of the given name is removed.

Exit Paths

Success

This path is taken if the schema is successfully added to the schema cache.

Failure

This path is taken if the operation fails.

XML Assign Node

This XML tool assigns the XML Node represented by 'Source Variable' to 'Destination Variable.'

Inputs

Source Variable

The node variable to assign to the destination variable.

Outputs

Destination Variable

The new node variable that represents to the same physical XML node as 'Source Variable.'

Exit Paths

Next

This path always takes the Next exit path.

XML Assign Node Iterator

This XML tool assigns the Node Iterator given by Source Variable to the Destination Variable. The iterator returned in Destination Variable points to the same sequence and same position in the sequence. However, the returned iterator is an iterator in its own right, and thus calling [XML Get Next Node](#) on it will not modify the position of the Source Variable iterator. This tool allows saving a certain position in the iteration sequence and come back to it at a later point in time. This tool thus performs assignment by *value* and not by *reference*.

Inputs

Source Variable

The node variable to assign to the destination variable.

Outputs

Destination Variable

The new node variable that represents to the same physical XML node as Source Variable.

Exit Path

Next

This path always takes the Next exit path.

XML Clone Node

This XML tool clones the node and, depending on the Recursive flag, all its descendants. If the node is not a Document, the resulting node will have the same owner document as the source node, but the new node will not have a parent. You can use the [XML Insert Node](#) tool to attach the new node or branch to a different document.

If the node is a Document, the whole document is duplicated and returned as a result. If the source node is a member of a read-only document, the node can only be cloned if Create New Document is True. In this case, the cloned document is not read-only.

Note: If the Source node is a document, the `SelectionLanguage` and `SelectionNamespaces` document properties are copied to the destination document as well.

Inputs

Source

Node to clone.

Clone Recursively

Set this to True to clone the node and all its descendants. Set to False to clone only the current node, along with its attributes, if applicable.

Note: does not apply to Document nodes. Documents are always cloned as a whole.

Create New Document

Set this to True to clone the Source node as a document element of a new document. Set to False to create the new node with no parent, but with the same owner document as the source node. The new document will not be read-only, even if the source document is.

Note: Source node must be an Element when this parameter is set to True.

Outputs

Clone

Exact clone of the node (and its descendants, if enabled).

Exit Paths

Success

This path is taken if the node was successfully cloned.

Failure

This path is taken if the operation fails.

XML Create Document

This XML tool creates a document from a prepackaged string resource stored in the handler file. The properties dialog of the tool allows editing the XML data. Files can be imported into the editor. The XML data can be checked for content and formatted in the editor.

Note: Even if data is imported from a file, the data is always stored locally in the handler. Thus, no reference is maintained to the file.

XML Data

This field displays the XML data being used to create the document.

Settings

Preserve Whitespace

Checking this box preserves white spaces when loading the document. This flag specifies the default white space handling when the `xml:space` attribute is set to "default." When the parameter is true, all whitespace is preserved, regardless of the `xml:space` settings in the document. When false, the values of the `xml:space` attribute specified in the document determine whether white space is preserved or not.

By default, this box is not checked.

Validate On Parse

Checking this box forces the validation of the document during parsing. By default, this box is not checked.

Resolve Externals

Checking this box causes the parser to resolve resolvable externals such as namespaces, DTD external subsets, and external entity references. By default, this box is not checked.

Cache Document

When this box is checked, the document is cached, allowing for very efficient re-use. This option is most useful for frequently used documents that do not have to be changed programmatically, most notably XSLT style-sheets. Enabling caching of style-sheets is particularly efficient, as pre-compiled templates of the style-sheets are kept in the cache.

Cached documents may be evicted from the cache if they are not used for more than about 10 minutes. However, this is completely transparent, as a new instance will be created and cached the next time a the document is created.

If the box is not checked, a new instance of the document will be parsed from the string resource in the handler every time a handler invokes the tool.

This box is checked by default.

Read Only

When this box is checked, the XML Document tree cannot be modified. An error occurs when attempts are made to modify a read-only XML document. This box is checked by default.

XML Document

The node of the document. The value is NULL if an error other than parse error occurred.

XML Document Element

The root element of the document. The value is NULL if the document has no root element or if a (parse) error occurred.

Selection Namespaces

This list associates a prefix with the corresponding namespace URI for XPath selection operations. See description for `SelectionNamespaces` property in [XML Set Document Property](#) tool for more details. The four buttons on the bottom of the field allow you to manage this list.

Add

Clicking this buttons opens a dialog box that allows you to add a new item to the list. Once the dialog box is open:

1. Select a namespace URI from the dropdown list or enter a namespace URI. The dropdown list shows a list of all namespaces used by the data currently in the edit box. However, you are not required to use these namespace URIs, this is just for convenience.
2. Enter the prefix to be associated with the namespace URI. The dropdown list shows the prefix of the first namespace declaration using the given namespace URI. Again, this is just for your convenience, as one usually likes to use the same prefixes in the selection patterns as the appear in the document. However, you may specify any prefix you like.

Note: Prefix names must be unique, but you may declare an arbitrary number of prefix mappings for the same namespace URI.

Edit

This button allows you to edit an existing entry on the list.

Remove

This button deletes the selected item from the list.

Clear

Clicking this button deletes all entries from the list.

Exit Paths

Success

This path is taken if the document is successfully parsed.

Parse Error

This path is taken if an error occurs parsing the data. Use the [XML Get Error Info](#) tool with the Document node as the argument to retrieve rich information about the error.

Failure

This path is taken if the operation fails for any reason other than a parsing error.

XML Create Document From String

This XML tool parses the Data string as XML and returns a document representing the XML data

IMPORTANT: The XML string must contain one top-level element (root). The `<?xml version=1.0" ?>` may be omitted. Thus, the simplest valid XML document (apart from an empty document) is `<x/>`.

Inputs

String Data

The data string to parse. The data string must contain valid XML to prevent parsing errors.

Preserve Whitespace

Specifies whether whitespaces in the document are preserved. This flag specifies the default white space handling when the

xml:space attribute is set to "default." When the parameter is true, all white space is preserved, regardless of the xml:space settings in the document. When false, the values of the xml:space attribute specified in the document determine whether white space is preserved or not.

Validate On Parse

Setting this argument to True forces the validation of the document during parsing.

Resolve Externals

If this parameter is True, resolvable externals such as namespaces, DTD external subsets, and external entity references are resolved when the document is parsed.

Default Selection Namespaces

A string containing a whitespace-separated list of namespace prefix declarations to be used for XPath selection tools.

This is the same data as can be set using the `SelectionNamespaces` document property (see [XML Set Document Property](#)). Thus, this argument acts like invoking the XML Set Document Property tool right after this tool. However, you have to specify the selection namespaces here, in particular for read-only documents, as calling XML Set Document Property on read-only documents is not permitted.

Read Only

When this box is checked, the XML Document tree cannot be modified. An error occurs when attempts are made to modify a read-only XML document. The box is unchecked by default.

Outputs

Document

The node of the document. The value is NULL if error other than parse error.

Document Element

The root element of the document. The value is NULL if document has no root element or an error occurs.

Exit Paths

Success

This path is taken if the document is successfully parsed.

Parse Error

This path is taken if an error occurs while parsing the data. Use the [XML Get Error Info](#) tool with the "Document" node as the argument to retrieve information about the error.

Failure

This path is taken if the operation fails for any reason other than a parsing error.

XML Create Node

This XML tool creates a new node and optionally appends it as child of the Parent Node. The node is appended to the parent as described below:

Element

Parent node must be an Element or Document node. If it is an Element node, the new node is appended to the child list of the element. If it is a Document, the new node *replaces* the document element (root element).

Attribute

Parent node must be an Element. Node is added to the attributes of the element. If an attribute of the same name already exists, it is replaced.

Text, CDATA

Parent node must be an Element. Node is appended to the child list of the element.

Comment, Processing Instruction

Parent node must be an Element or Document. Node is added to the child list of the parent.

If a new node is created without a parent, the node will be a member of an empty dummy document just created for the node. Thus, the 'Owner Document' output of the [XML Get Node Info](#) tool always returns a document. Once the node is inserted into the document tree of a different document, the dummy document is destroyed.

Note: If creating the node fails, the output node is a document node (irrespective of the 'Node Type') that can be queried with XML Get Error Info for detailed error information. This document node is either the owner document of 'Parent Node', or, if no parent node is given, an empty dummy document. If something is seriously wrong and creating the document node failed, 'Node' will be NULL.

Inputs

Parent Node

Optional. Node to which to add this node as child. Semantics depend on the node type. If no parent node is given, the node may be added using the XML Insert Node tool. See description above.

Node Type

Integer value corresponding to the type of node to create. For list of values see [XML Get Node Info](#) tool. Nodes of type *Document*, *Document Type*, *Entity*, or *Notation* cannot be created.

Name

Fully qualified name of the node. Should be empty for nodes that have no name (*CDATA*, *Comment*, *Text*, *Document*, *Document*, *Document Fragment*).

Namespace URI

Namespace unique resource indicator (URI). If specified, the node is created in the context of this namespace with the prefix specified on the node name. If the Name parameter does not have a prefix, a default namespace declaration will be inserted as appropriate. This applies only to *Element* and *Attribute* nodes.

Node Value

Value of *Attribute*, *Text*, *CDATA*, *Comment*, and *Processing Instruction (PI)* nodes. If this argument is non-empty and the node is an *Element*, a text node is added as child of the Element containing that text. If a data type is defined, the node value must be parseable as that value.

May be left empty and then populated with the [XML Set Node Value](#) tool.

Data Type

String representation of the data type specifier included in the schema. Empty string: type undefined.

Ignored for node types that don't support it.

This is the type defined as the nodes dt:dt attribute (dt is "urn:schemas-microsoft-com:datatypes" namespace)

Outputs

Node

Newly created node

Exit Paths

Success

This path is taken if the node was successfully created.

Failure

This path is taken if the operation fails.

XML Escape Entities

This XML tool returns a string with all occurrences of the specified entity characters replaced with the corresponding entity reference.

Note: Stricter escaping has precedence. Thus, if 'Escape non-ASCII Characters' and 'Escape non-ANSI Characters' check boxes are selected, the ASCII escaping takes precedence, as it is stricter.

Inputs

Source String

String to process.

Escape < with <

Checkbox, default = True. Escape < to `<`;

Escape > with >

Checkbox, default = True. Escape > to `>`;

Escape & with &

Checkbox, default = True. Escape & to `&`;

Escape ' with '

Checkbox, default = True. Escape ' to `'`;

Escape " with "

Checkbox, default = True. Escape " to `"`;

Escape non-ASCII Characters

Checkbox, default = True. Escape characters that are not characters from the basic ASCII character set (0x20...0x7E). The control characters supported by XML (`\t`, `\n`, `\r`) are not escaped.

Escaped characters are replaced by a hexadecimal character reference (`&#x<code>`).

Escape non-ANSI Characters

Checkbox, default = False. Escape characters that are not characters from the ANSI character set (0x20...0xFF). The control characters supported by XML (`\t`, `\n`, `\r`) are not escaped. Escaped characters are replaced by a hexadecimal character reference (`&#x<code>`).

Normalize LF to CR/LF

Checkbox, default = False. Replace newlines (`\n`) with carriage-return/newline pairs (`\r\n`). Already present CR/LF pairs will not be expanded (i.e. CR/LF will not be expanded to CR/CR/LF)

Normalize CR/LF to LF

Checkbox, default = False. Replace carriage-return/newline pairs (`\r\n`) with a linefeed (`\n`).

Output

Result String

String with certain characters escaped.

Exit Path

Next

This path always takes the Next exit path.

XML Get Attribute

This XML tool retrieves an attribute node and its value from an element node by attribute name.

The name of the attribute to retrieve can be given as qualified name (`xxx:yyy`) or as Base Name (`yyy`) Namespace URI pair. They are mutually exclusive, and the failure path is taken if both a Name and a Base Name/URI pair are given.

Note that Name = "yyy" is the same as Base Name = "yyy" and 'Namespace URI' = "".

Inputs

Node

Node that's attributes are to be queried. Must be Element.

Name

Qualified Name of the attribute to retrieve. This field should be left empty if 'Base Name' and 'Namespace URI' are to be used, as both of those fields will be ignored if 'Name' is given a value.

Base Name

Base name of the attribute to retrieve. This field will be ignored if 'Name' has a value specified.

Namespace URI

Namespace of the attribute to retrieve. This field will be ignored if 'Name' as a value specified.

Outputs

Attribute Node

Attribute node of the attribute of the element. NULL if no attribute of the given name.

Attribute Value

String containing the value of the attribute.

Exit Paths

Success

This path is taken if the attribute was successfully retrieved.

Unknown

This path is taken if the element has no attribute of the given name.

Failure

This path is taken if the node is not an element or some other unexpected error occurs.

XML Get Attributes

This XML tool returns an iterator pointing to the first attribute of the list of attributes of an element. Use [XML Get Next Node](#) to retrieve the node at the current position and advance the iterator.

Note: The collection retrieved by this tool is "live." See description in [XML Get Child Nodes](#).

Inputs

Node

Node of an Element whose attributes are to be retrieved. This node must be Element.

Outputs

Attribute Nodes

Iterator pointing to first item in the list of attributes of the element.

Exit Paths

Success

Child nodes were successfully retrieved.

Empty

Specified element has no attributes..

Failure

This path is taken if the operation fails.

XML Get Child Nodes

This XML tool returns an iterator pointing to the first item of the list of child nodes of 'Node'. The tool [XML Next Node](#) can be used to retrieve the node at the iterator position and advance to the next node. The following node types may have children: *Element*, *Attribute*, *Document*, *Entity*, *Entity Reference*, *Document Fragment*, *Document Type*.

Notes: Attributes are not children of an element. You have to use the [XML Get Attributes](#) tool to retrieve a list of attributes of an Element.

The collection to which the node iterator points is "live." Thus, adding or removing child nodes to or from 'Node' while iterating over the child nodes will be reflected immediately in the collection. The iteration is always stable (thus, if you insert a child after the current iteration position, the iteration will hit that child when stepping to the corresponding position. Children inserted before the current position will not be visited. If you restart the iteration at the beginning, for example by retrieving an iterator to the first position with 'XML Get First Iterator Position', all children will be visited, even the subsequently inserted ones.

The same is true to removal of children, even the child to which the iterator points might be removed. The iterator will visit the corresponding next child in the sequence.

Inputs

Node

Node whose child nodes to retrieve.

Outputs

Child Nodes

Iterator to iterate over the list of child nodes. Points to first item in the list.

Exit Paths

Success

Child nodes successfully retrieved.

Empty

The node has no child nodes.

Failure

This path is taken if the operation fails.

XML Get Document Property

This XML tool queries the value of a document property. See [XML Set Document Property](#) for a list of the supported properties.

Inputs

Document

A node in a document that contains the property to query.

Property Name

The string containing the name of the property to retrieve. Property names are case sensitive.

Outputs

Property Value

The value of the property.

Exit Paths

Success

This path is taken if the document properties were successfully retrieved.

Failure

This path is taken if the operation fails.

XML Get Error Info

This XML tool retrieves the parse error data associated with the given document or information about the last error that occurred on that node. If there is no parse error in the document or the last command was executed successfully, the 'No Error' path is taken. If the node is not a document node, the owner document will be queried for a parse error instead.

The string returned by 'Formatted Error Position' can be used for simple error reporting, such as in trace or event logs.

This tool does not change the error status of the Node.

Note: Passing a NULL Node will be reported as a dummy error ("XML Node is NULL!").

Inputs

Node

The node of the document that had a parse error, or a node passed to a tool when the failure path is taken.

Outputs

Error Code

Native error code (parse error code or HRESULT). 0 à no error

File Position

Absolute character position in the file (or string) where the error occurred. -1 if no parse error.

Line

Line number where the error occurred. -1 if not a parse error.

Line Position

Character position in the current line. -1 if not a parse error.

Reason Text

Descriptive text of the error.

Source Data

Data of the line containing the error or module that caused the error ('GetSource' field of IErrorInfo).

URL

URL of the document (empty for document created from strings or if not a parse error).

Formatted Error Position

Two lines of text separated by a newline (\n). The first line contains an excerpt of the source line and the second a marker where the error is. Thus the string will contain something like:

```
<foo attr="abc">Test</bar>\n
_____^
```

This will be empty if there is not a parse error.

Exit Paths

Success

This path will be taken if the error information is successfully retrieved.

No Error

This path will be taken if there was no error information.

Failure

This path will be taken if the operation fails.

XML Get First Iterator Position

Given a node list iterator pointing to any location in a node list, this XML tool returns an iterator that points to the first location in that node list.

Note: This tool does not modify the iterator given as an argument. Instead, it returns a new iterator pointing to the first item of the same collection as the argument. If you specify the same variable as an output argument that you also specified as input, the iterator will effectively reset back to the first position.

Inputs

Node Iterator First

Node list iterator, may point to any location in node list.

Outputs

Node Iterator

Node list iterator pointing to first location in the list.

Count

Number of nodes in the node list.

Exit Paths

Success

This path is taken if the child nodes were successfully retrieved.

End

This path is taken if the node output retrieves a value of NULL, indicating that iteration is complete.

Failure

This path is taken if the operation fails.

XML Get Namespaces

This XML tool returns a list of strings containing all namespace URIs used in the document that owns the given node. Thus, all namespace URIs used in the document are always returned, even if the node is some descendant of the document node.

Inputs

Document

Node of document whose namespaces are to be returned.

Outputs

Namespace URI List

List containing strings of the namespace declaration URIs used in the document.

Exit Paths

Success

This path is taken if the namespaces are successfully retrieved.

Empty

This path is taken if there are no namespace URIs in the document.

Failure

This path is taken if the operation fails.

XML Get Next Node

This XML tool returns the node at the current iterator position and returns an iterator pointing to the next position in the list. As the iterator is just a variable, you can make copies at any time to save a certain position. By using the same variable as input and output iterator, you can easily iterate over the list by connecting the Success path back to this tool (after processing the node, of course).

The tool takes the 'End' exit when the iterator points to an empty list or the iteration is complete (list traversed to end).

Inputs

Node Iterator

Current position in the node list.

Outputs

Node Iterator Next

Iterator pointing to the next position in the list.

Node

Node at the current position. NULL if list is empty.

Exit Paths

Success

Successfully retrieved node at the current position.

End

Iteration complete.

Note: This path is taken if this tool is invoked again after the last item has been retrieved. The 'Node' output variable is set to a NULL node.

Failure

This path is taken if the operation fails.

XML Get Node Info

This XML tool retrieves information about the node passed as an argument.

Inputs

Node

The node to be queried.

Outputs

Name

Qualified name for the *Element*, *Attribute*, or *Entity Reference*. For example, it returns `xxx:yyy` for the Element `<xxx:yyy>`.

Base Name

Right side of a namespace qualified name. For example, it returns `yyy` for the element `<xxx:yyy>`.

Prefix

Namespace prefix specified on the *Element*, *Attribute*, or *Entity Reference*. For example, for the element `<xxx:yyy>`, it returns `xxx`. It returns an empty string, "", if no prefix is specified

Namespace URI

Universal resource identifier (URI) for the namespace. This refers to the `uuu` portion of the namespace declaration `xmlns:nnn="uuu"`.

Node Type

Integer value corresponding to the type code of the node. Valid type codes are:

- 1 Element
- 2 Attribute
- 3 Text
- 4 CDATA Section
- 5 Entity Reference
- 6 Entity
- 7 Processing Instruction (PI)
- 8 Comment
- 9 Document
- 10 Document Type
- 11 Document Fragment
- 12 Notation

Node Value

Value of **Attribute**, **Text**, **CDATA**, **Comment**, and **Processing Instruction (PI)** nodes. Empty string for all other nodes.

Note: For Element nodes, this is not the same value as returned by 'XML Get Node Value'! Elements themselves have no value, and thus an empty string is returned.

Data Type

String representation of the data type specifier included in the Schema for this node (Element or Attribute). The string is empty if no type is specified.

Parent Node

Parent node of the node. NULL if node has no parent (Document and Attribute nodes, for example)

Owner Document

Document of this node.

Definition

Definition of this node in the DTD or XML Schema. NULL if no associated schema definition.

Exit Paths

Success

This path is taken if the node information was successfully retrieved.

Failure

This path is taken if the operation fails.

XML Get Node Value

This XML tool retrieves value of a node. This operation is only allowed on Element, Attribute, Text, CDATA, Comment and Processing Instruction nodes. All other nodes cause an error.

If the Node is an *Element*, the returned value corresponds to the typed value as defined by the data type (`dt:dt` attribute, data type defined in the Schema) of the Element, coerced into a String. If the element has mixed contents, the returned value consists of a concatenation of the node values of all child nodes. This special functionality for element nodes is particularly convenient for typed nodes. For example, passing the following "FOO" node to the 'XML Get Node Value' tool will return a string containing "-12345" (we assume here that `dt` is the prefix for "urn:schemas-microsoft-com:datatypes"):

```
<FOO dt:dt="int">-12345</FOO>
```

Inputs

Node

The node that's value is to be changed.

Outputs

Node Value

String representing the value associated with the node.

Exit Paths

Success

This path is taken if the node value was successfully retrieved.

Failure

This path is taken if the operation fails.

XML Get Schema

This XML tool retrieves the Schema of the given URI. It returns the node of the <Schema> tag, when the schema is an inline schema, or the schema is an external schema and the document was loaded with 'Resolve Externals' turned on.

This tool first checks the schema cache of the document for schemas of that namespace. If none are found, the namespace collection is checked. The namespace collection contains the namespaces used in the document and any schemas that are associated with it. The schema cache contains schemas added manually through the [XML Add Schema](#) tool.

Note: The returned Schema node is always read-only.

Inputs

Document

Node of the document to query for a Schema.

Namespace URI

Namespace URI of the Schema to retrieve.

Outputs

Schema

Node of the Schema associated with the given URI. NULL of no Schema or unknown URI.

Exit Paths

Success

This path is taken if the schema is successfully retrieved.

Unknown URI

This path is taken if the given URI is neither used in the document nor in the schema cache.

No Schema

The namespace URI is known, but there is no schema loaded for it.

Failure

This path is taken if the operation fails.

XML Get Sibling

This XML tool returns the next or previous sibling of Node in document order. If there is no such sibling, a NULL node is returned and the No Sibling exit is taken.

Inputs

Node

Node that's sibling is to be returned.

Previous Sibling

If this box is checked, the previous sibling node in document order will be returned. Left unchecked, the next node in document order will be returned. The returned value will be NULL if the appropriate sibling does not exist. The box is unchecked by default.

Outputs

Sibling Node

Next or previous sibling of the node. NULL node if node has no previous/following sibling.

Exit Paths

Success

This path is taken when the sibling node has been retrieved successfully.

No Sibling

This path is taken if node has no following or previous sibling. A NULL node is returned.

Failure

This path is taken if the operation fails.

XML Get Text

This XML tool returns the text of the node. The text returned depends on the type of node as follows:

Attribute, Document, Entity:

Returns a string representing the value of the node. This comprises the concatenated node value of all its child nodes with entities expanded.

Text, CDATA, Comment, Processing Instructions:

Returns the text contained in the node, which is the same as 'Value' returned by XML Get Node Info.

Element:

Returns a string that represents the element content. Note that this will also include the text content from all child elements, concatenated in document order. For example, the branch `<A>ThisIsATest` will return "ThisIsATest".

Document Fragment:

Returns the text comprised of the concatenation of all descendant nodes.

Entity Reference:

Returns the string representation of the entity reference.

Document Type, Notation:

Returns the empty string "". These node types do not have associated text.

Inputs

Node

Node whose text is to be returned.

Outputs

Text

Text of the node and potentially of all its child nodes concatenated, entity references expanded etc., depending on type of the node. See the description above.

Exit Paths

Success

This path is taken if the node text was successfully retrieved.

Failure

This path is taken if the operation fails.

XML Get XML

This XML tool creates a string that contains the XML representation of the node and all its descendant nodes.

Inputs

Node

Node that's XML representation is to be extracted.

Outputs

XML String

XML representation of the node and its descendants.

Exit Paths

Success

This path is taken if the XML representation was successfully retrieved.

Failure

This path is taken if the operation fails.

XML Insert Node

This XML tool appends the New Child node to Node, or inserts New Child before Insert Before in the child list of Node. The types of nodes allowed as New Child depend on the type of Node.

Node types not listed below may not be specified as Node:

Attribute

Allowed child node types: *Text, Entity Reference, and Document Fragment.*

Document

Allowed child node types: *Comment, Processing Instruction, Element, Document Fragment.*

If the New Child node is an Element and the document already has a document element (root element), the existing document element will be replaced (Insert Before is ignored).

Document Fragment

Allowed child node types: *CDATA, Comment, Element, Text, Entity Reference, Processing Instruction, Document Fragment.*

Element

CDATA, Comment, Element, Text, Attribute, Entity Reference, Processing Instruction, Document Fragment.

Notes: This tool allows adding attributes to an element, even though attributes are not considered actual children of element nodes. This is a simplification to keep the number of tools low. If the New Child node is an attribute, it is added to the element. If an attribute of the same name as the added attribute already exists, the existing attribute is replaced with the new node. Insert Before must be left undefined.

This tool may not only be used to add new nodes but also to move nodes and whole branches in a document or to move nodes or branches to a different document. The node parent and owning document will be adjusted as necessary. If you want to move nodes within a document or between documents, you don't need to remove the node before inserting it in the new location.

Errors occurring while inserting the node will be attached to Node. Thus, you have to call [XML Get Error Info](#) with Node as argument.

Inputs

Node

Node to which to add the new child node (new parent of New Child).

New Child

Node to add to Node.

Insert Before

Optional. Child node before which to insert the new child. If not specified, node will be added to end of the child list.

Exit Paths

Success

This path is taken if the node was successfully inserted.

Failure

This path is taken if the operation fails.

XML Load Document

This XML tool loads the document as specified by the URL. The document is loaded synchronously. Thus, the tool blocks until the document is loaded and parsed completely. Note that the document may contain external DTD or Schema references that will be resolved too.

Inputs

URL or File Name

URL or file name of the document to load. For example, 'C:\text.xml' or '\\test\c\test.xml'.

Preserve Whitespace

Specifies whether whitespaces in the document are preserved. This flag specifies the default white space handling when the `xml:space` attribute is set to "default." When the parameter is true, all white space is preserved, regardless of the `xml:space` settings in the document. When false, the values of the `xml:space` attribute specified in the document determine whether white space is preserved or not.

Validate On Parse

Setting this argument to True will force the validation of the document during parse.

Resolve Externals

If this parameter is True, resolvable externals such as namespaces, DTD external subsets, and external entity references are resolved when the document is parsed.

Default Selection Namespaces

String containing a whitespace separated namespace prefix declarations to be used for XPath selection tools.

This is the same data as can be set using the `SelectionNamespaces` document property (see [XML Set Document Property](#)). Thus, this argument acts like invoking the XML Set Document Property tool right after this tool. However, you have to specify the selection namespaces here, in particular for read-only documents, as calling XML Set Document Property on read-only documents is not permitted.

Cache Document

If this box is checked, the document is cached based on the document URL (case sensitive!) and modification time. Cached documents are always read-only, irrespective of the 'Read Only' parameter.

The following parameter 'Preserve Whitespace', 'Validate On parse', 'Resolve Externals', and 'Default Selection Namespaces' are also used to identify the document in the cache. If the URL is a filename, the modification time is used too. Thus, after the file is modified, the next time the tool is invoked, a new document for the file is created and added to the cache. The previous document stays in the cache and is reclaimed if not accessed for about 10 minutes and no references to it are outstanding in other handler instances. Apart from making subsequent loads of the same document very efficient, caching has the advantage that documents that represent XSLT style-sheets are compiled and cached in a pre-compiled form the first time an XSLT function is invoked, so subsequent transformations with the same style-sheet are very efficient.

NOTE: Enabling document caching should not be used for documents that are changed frequently or very large documents, as the memory footprint of the process could be unacceptably inflated.

Read Only

When this box is checked, the XML Document tree cannot be modified. An error occurs when attempts are made to modify a read-only XML document. The box is unchecked by default.

Outputs

Document

The node of the document. The value is NULL if error other than parse error.

Document Element

The root element of the document. The value is NULL if document has no root element or an error occurred.

Exit Paths

Success

This path is taken if the document is successfully parsed.

Parse Error

This path is taken if an error occurs while parsing the data. Use the [XML Get Error Info](#) tool with the "Document" node as the argument to retrieve information about the error.

Failure

This path is taken if the operation fails for any reason other than a parsing error.

XML Remove Node

This XML tool removes the designated node from the children of Node. If Node is an element and the child is an attribute, the attribute is removed, even though attributes are not considered children of an element.

Inputs

Node

Node from whose children to remove a node.

Child To Remove

Child to remove.

Exit Paths

Success

This path is taken if the node was successfully removed.

Failure

This path is taken if the operation fails.

XML Save Document

This XML tool saves the document of the node to a file. Note that even if the node is somewhere down in the tree, the whole document is always saved. To extract only a branch of the tree, use the [XML Get XML](#) tool on the root node of the branch to export and write the returned string to a file. Alternatively, you can clone the node and its children into a new document and save that document.

Inputs

Document

The node of the document to be saved.

Filename

Filename for the saved document.

Exit Paths

Success

This path is taken if the document is successfully written to file.

Access Denied

This path is taken if the save operation on the destination file not permitted.

Failure

This path is taken if the operation fails for any other error.

XML Select Nodes

This XML tool evaluates the given XPath pattern against the context as defined by the Context pattern and returns an iterator to the resulting list of nodes.

Note that the Context node only defines the current context. You can still navigate the whole document tree. Thus, the pattern "/" always returns the document node, even if the context is not the document node.

Note: Please be aware of the XPath rules for names with namespace prefixes. Element names in XPath expressions have to be qualified with a namespace prefix defined in the `SelectionNamespace` property, even if the element is in the scope of a default namespace and has no prefix. For example, given the following XML document:

```
<A xmlns="uri:some-namespace">
<B>FOO</B>
</A>
```

The XPath query `/A/B` will *not* select element B! You will instead have to specify a prefix using the `SelectionNamespaces` property and use this prefix in the query.

Example:

```
SelectionNamespaces: "xmlns:x="uri:some-namespace"
XPath query: "/x:A/x:B"
```

For some additional details, please consult the description of the `SelectionNamespaces` property in the [XML Set Document Property](#) tool.

Inputs

Context

Context from which to evaluate the XPath pattern.

Pattern

XPath pattern describing nodes to select.

Outputs

Node List

Iterator pointing to first item in the node list resulting from evaluating the given XPath expression against the context defined by the Context node.

Exit Paths

Success

This path is taken if the nodes were successfully selected.

No Match

This path is taken if no nodes match the given XPath query. XPath matches the context as defined in the Context input.

Failure

This path is taken if the operation fails.

XML Select Nodes As List

This XML tool evaluates the given XPath pattern against the context as defined by the 'Context' node and returns a list of strings, where each item contains the node value of one of the selected nodes. The order in the list corresponds to the order of nodes selected by the XPath query.

Note that the 'Context' node only defines the current context. You can still navigate the whole document tree. Thus, the pattern "/" will always return the root node of the document, even if the context is not the root node.

NOTE: The namespace prefixes that can be used in XPath queries must be set using the `SelectionNamespaces` document property (see [XML Set Document Property](#)) or can be passed as argument with the document creation tools.

Inputs

Context

The context from which to evaluate the XPath pattern.

Pattern

The XPath pattern describing nodes to select.

Outputs

Node Values

The List of string containing the node values of the query result.

Exit Paths

Success

This path is taken if the nodes were successfully selected.

No Match

This path is taken if no nodes match the given XPath query. XPath matches the context as defined in the Context input.

Failure

This path is taken if the operation fails.

XML Select Single Node

This XML tool evaluates the XPath expression and returns the first node that satisfies the pattern. It can also return the node value, which makes this tool particularly useful for retrieving the data of an arbitrary node in a document hierarchy. The node value is the same as retrieved by [XML Get Node Value](#), thus, if Retrieve Value is checked, this tool acts like if the matching node was fed into the [XML Get Node Value](#) tool.

See the [XML Get Node Value](#) tool for semantics of the node value.

Note that the Context node only defines the current context. You can still navigate the whole document tree. Thus, the pattern "/" always returns the document node, even if the context is not the document node.

Inputs

Context

Context from which to start evaluating the XPath pattern.

Pattern

XPath pattern to evaluate in the context.

Retrieve Value

True Select node and return handle to the node and retrieve value of the selected node. Should only be turned on if node value is of interest, can be quite expensive if the node is an element with many descendants.

False Only select the node and return a handle to the node. Node Value is empty. The check box is set to False as the default.

Outputs

Matching Node

First node resulting from evaluating the XPath expression. NULL if no node satisfies the pattern.

Node Value

Node Value of the selected node. Empty if no node satisfies the pattern or if Retrieve Value is false.

Exit Paths

Success

This path is taken if a node matched the XPath query.

No Match

This path is taken if no node matches the XPath query.

Failure

This path is taken if the operation fails.

XML Select Single Node Set Value

This XML tool evaluates the XPath expression and changes the node value of the first node that satisfies the pattern. This is a convenience tool that simplifies the use of XML as structured data. It basically wraps a call to XML Select Single Node and XML Set Node Value into one tool.

Please consult description of [XML Set Node Value](#) tool for a description of the semantics of the value, in particular in conjunction with data types.

Inputs

Context

Context from which to start evaluating the XPath pattern

Pattern

XPath pattern to evaluate in the context

New Node Value

Value to set as the new nodes value. Must match the data type, if data type is defined for node.

New Data Type

Integer value corresponding to the new data type to be set before setting the value. See the [XML Get Node Info](#) tool for the list of valid data types. If a type is not specified, the type will not be changed.

Outputs

Matching Node

Node whose value has been changed. NULL if no node satisfies the pattern.

Exit Paths

Success

This path is taken if a node matched the XPath query and the value has been successfully changed. the child nodes were successfully retrieved.

No Match

This path is taken if no Node matches the XPath query.

Failure

This path is taken if the operation fails, for example if the value does not agree with the data type of the node.

XML Set Attribute

This XML tool changes the value of the named attribute. The node must be an element; otherwise the tool will take the 'Failure' exit. If the element has no attribute of the given name, the tool returns through the 'Unknown' exit.

Inputs

Node

Element node with attributes to change.

Name

Qualified Name of the attribute to retrieve. This field should be left empty if Base Name and Namespace URI are to be used, as both of those fields will be ignored if Name is given a value.

Base Name

Base name of the attribute to retrieve. This field will be ignored if Name has a value specified.

Namespace URI

Namespace of the attribute to retrieve. This field will be ignored if Name as a value specified.

Attribute Value

String to set as new value of the attribute. If a data type is specified for the attribute (e.g. through the Schema, the string must be coercible into that value)

New Data Type

Integer value corresponding to the new data type to be set before setting the value. See the [XML Get Node Info](#) tool for the list of valid data types. If a type is not specified, the type will not be changed.

Create if Unknown

This checkbox determines the tool's behavior when it is directed toward an attribute that does not presently exist. If set to True, a new attribute will be created with the given 'Name' value and Data Type, and it will be added to the Element.

If set to False, the tool will exit through the Unknown exit path.

Outputs

Attribute Node

Attribute node that has been modified. NULL if no attribute of the given name.

Exit Paths

Success

This path is taken if the attribute was successfully retrieved.

Unknown

This path is taken if the element has no attribute of the given name.

Failure

This path is taken if the node is not an element.

XML Set Document Property

This XML tool modifies the value of a document property. The following properties are supported:

Name:	Description:
SelectionLanguage	Specifies the selection language for the 'XML Select xxx' operations. XPath Use XPath as selection language. Default XSLPattern Use Microsoft XSL Patterns as selection language NOTE: The default is different from the default of the MSXML parser, as XPath is the way of the future!
SelectionNamespaces	Specifies the selection namespaces used for XPath queries. Please see the extended description following this table.

The **SelectionNamespaces** property allows you to specify a white-space separated list of namespace declarations of the form `xmlns:<prefix>=<namespace-uri>`. These namespaces define the prefixes for XPath selection operations, such as [XML Select Nodes](#). In XPath queries, namespace prefixes are **NOT** the same as the ones appearing in the document. For example, given the document:

```
<a:A xmlns:a="urn:test-a">  
<b:B xmlns:b="urn:test-b">Foo</b:B>  
</a:A>
```

Neither the XPath pattern `/A/B` nor `/a:A/b:B` will select element B. This might appear counter-intuitive, but has a good reason: the namespace prefix is not necessarily unique, as elements in the same scope can have the same prefix for different namespace URIs, such as in the following document:

```
<n:A xmlns:n="urn:test-a">  
<n:B xmlns:n="urn:test-b">Foo</n:B>  
<n:B xmlns:n="urn:test-c">Bar</n:B>  
</n:A>
```

There would be no way to specifically select the second B element based on its namespace. Thus, the namespace prefixes for XPath selections have to be defined separately. The document property **SelectionNamespaces** provides a way to specify the namespace prefixes to be used in the XPath operations. Thus, specifying the string `xmlns:x='urn:test-a' xmlns:y='urn:test-b' xmlns:z='urn:test-c'` as selection namespaces allows to use the XPath query `/x:A/y:B` to retrieve the first B element and `/x:A/z:B` to retrieve the second B element in the above document.

NOTE: The [XML Create Document](#), [XML Create Document From String](#) and [XML Load Document](#) have a parameter to pass a selection namespace declaration. This basically provides the same functionality as loading the document and then calling 'XML Set Document Property' for the **SelectionNamespaces** property (but it also allows setting the selection namespaces for read-only documents).

Inputs

Document

Node from whose owner document to query a property.

Property Name

String containing the name of the property to retrieve. Property names are case sensitive.

Property Value

String containing the value of the property.

Exit Paths

Success

This path is taken if the document properties were successfully set.

Failure

This path is taken if the operation fails.

XML Set Node Value

This XML tool sets the value of a node. This operation is only allowed on *Element*, *Attribute*, *Text*, *CDATA*, *Comment* and *Processing Instruction nodes*. All other nodes cause an error.

If the node is an *Attribute* the value must be coercible into the data type defined for that node (either with 'XML Create Node' or in the Schema defining the attribute). If the type is not coercible, an error is issued.

If the node is an *Element* the value must be coercible into the data type set for that Element ('Data Type' in 'XML Create Node', `dt:dt` attribute of the element, or data type defined in the Schema definition). If the type is not coercible, an error is issued.

If the element has child elements, all child elements are replaced by one text node with the value.

If the Element is empty, a new text node with the value is added as the only child.

Example:

Given the following node (we assume here that `dt` is the prefix for "urn:schemas-microsoft-com:datatypes"):

```
<FOO dt:dt="int">-12345</FOO>
```

Passing the node and "abcd" as argument will cause an error because "abcd" cannot be parsed into a number. If no data type is defined for that node, the call succeeds, setting "abcd" as the new contents of the text child node.

NOTE: If the node is an empty element, a text node child is added.

Inputs

Node

The node with the value to be changed.

Node Value

The string to set as the node value.

New Data Type

This is the new data type to set before setting the value. See the [XML Get Node Info](#) tool for a list of valid data types. If the data type is not specified, the type will not be changed. If the string is left empty, the `dt:dt` attribute is removed.

Exit Paths

Success

This path is taken if the node value was successfully set.

Failure

This path is taken if the operation fails.

XML Switch On Node Type

This XML tool determines the node type and takes a different exit path depending on the type of the node. Only the more common node types are handled here. To determine the type of the less common nodes such as Entity Reference, Entity, Document Type, Document Fragment, or Notation, use the 'Node Type' result of the [XML Get Node Info](#) tool.

Inputs

Node

Node which is to have its type determined.

Exit Paths

Element

This path is taken if the node is an Element.

Attribute

This path is taken if the node is an Attribute.

Text

This path is taken if the node is a Text node.

CDATA

This path is taken if the node is a CDATA Section.

Document

This path is taken if the node is a Document.

Comment

This path is taken if the node is a Comment.

PI

This path is taken if the node is a Processing Instruction.

Other

This path is taken if the node is any node other than one previously listed.

NULL

This path is taken if the node handle is NULL.

Failure

This path is taken if an error occurred determining node type.

XML TRANSFORM TO DOCUMENT

This XML tool processes the given node and its children by applying an XSLT style-sheet. The transformation result is returned as XML document node. The Source node defines a context for the style-sheet to operate on, but navigation outside this scope is allowed. For instance, a style-sheet could use the ancestor or id methods to access other parts of the source document.

This tool allows passing parameters to the style-sheet. Two kind of parameters are supported:

- *String parameters* are defined as two lists whereas one contains the parameter names and the other the corresponding values. A namespace URI identifies the parameters in the style sheet. For the use of the String Parameters, consult the example below.
- *Node Parameters* are a list of XML nodes. These nodes are accessible through the DOM interface in the style-sheet. Each node is identified by a namespace URI supplied in a list of strings ('Node Parameter Namespace URIs'), where each item specifies the namespace URI of the node in the same position in 'Node Parameter List'.

To increase flexibility even further, the start mode of the style-sheet can be specified through the 'Start Mode' argument. Specifying the start mode can be useful to render different kinds of result documents based on some external condition, such as a debug mode.

The stylesheet does not have to be a document node, it may be any element node, as long as it is the root of a valid (X)SLT stylesheet. This allows, for example, to use documents with embedded stylesheets and use a selection step to pick a stylesheet to use.

Note 1: If the stylesheet node is not a document node or root element of a document, the stylesheet template will never be cached in a pre-compiled form (see XML Create Document or XML Load Document). It is therefore recommended to define frequently used stylesheets as separate documents to benefit from the performance improvement of cached, pre-compiled stylesheet templates.

Note 2: Even if an error occurs during the transformation, a document is returned in 'Document'. This document will most likely be empty, but can be passed to [XML Get Error Info](#) to retrieve verbose information about what went wrong. In most cases this error will be a style sheet error or a parse error of the resulting document (i.e. the resulting document is not well formed).

Note 3: The resulting document must obviously be well-formed XML in order to be returned as document. Care must be taken when HTML output is produced (`<xsl:output method="html"/>`), as some HTML tags do not constitute well formed XML (such as the META tag). If HTML output is to be produced, we recommend using the XML Transform To String tool instead.

Example: Source document:

```
<items>
<item id="1">This is Item 1</item>
<item id="2">This is Item 2</item>
<item id="3">This is Item 3</item>
<item id="4">This is Item 4</item>
</items>
```

Style-sheet:

```
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:params="urn:my-params"
exclude-result-prefixes="params"
version="1.0">
<xsl:output method="xml" indent="yes"/>
<xsl:param name="params:itemID"/>
<xsl:template match="/">
<result>
<xsl:text>Value: </xsl:text>
<xsl:value-of select="/items/item[@id=$params:itemID]"/>
</result>
</xsl:template>
</xsl:stylesheet>
```

String Parameter:

```
String Parameter Names: "itemID"
String Parameter Values: "2"
String Parameter Namespace URI: "urn:my-params"
```

Result:

```
<?xml version="1.0"?>
<result>Value: This is Item 2</result>
```

Inputs

Source

Node defining the context from which to start the transformation.

Stylesheet

Stylesheet to apply. This may be a document or an Element node representing a style sheet fragment.

String Parameter Names

Optional. List of names of the parameters passed to the XSLT style-sheet. Each item is paired up with the corresponding item in 'String Parameter Values'.

String Parameter Values

Optional. List of string values passed as parameters to the XSLT style-sheet. Each item is paired with the corresponding item in 'String Parameter Names'.

String Parameter Namespace URI

Optional. Namespace URI of the parameters passed to the XSLT style-sheet.

Node Parameter List

Optional. List of nodes to pass as parameters to the XSLT style-sheet. The nodes in the list are paired with the corresponding item in the 'Node Parameter Namespace URIs' argument.

Node Parameter Namespace URLs

Optional. List of namespaces that identify the corresponding object from the 'Node Parameter List' argument in the style-sheet.

Start Mode

Optional. Start mode for the XSLT transformation. This allows to specify a subset out of a larger XSLT style-sheet.

Start Mode Namespace URI

Optional. Namespace URI for the start mode. Allows to fully qualify the mode name.

Outputs

Result Document

Document node of the transformation result. A document node (usually empty) will be returned even if an error occurs during the transformation (unless the error is catastrophic). Pass to [XML Get Error Info](#) to retrieve verbose information.

Result Document Element

Root node (Element) of the document created as result of the transformation. NULL if error.

Exit Paths

Success

This path is taken if the document has been successfully transformed.

Failure

This path is taken if the operation fails.

XML Transform To String

This XML tool processes the given node and its children by applying the style sheet. The transformation result is returned as a string. The Source node defines a context for the style-sheet to operate on, but navigation outside this scope is allowed. For instance, a style sheet could use the ancestor or id methods to access other parts of the source document.

Note: If an error occurs during the transformation, error information is attached to the Source node. Thus, you have to invoke the [XML Get Error Info](#) tool with the Source node as argument to retrieve additional information. Please note the difference to the [XML Transform To Document](#), where the error information is returned in an empty document.

For a description and an example of style-sheet parameter arguments, please consult the [XML Transform To Document](#) tool.

Inputs

Source

Node defining the context from which to start the transformation.

Stylesheet

Style-sheet to apply. This may be a document or an Element node representing a style sheet fragment.

String Parameter Names

Optional. List of names of the parameters passed to the XSLT style-sheet. Each item is paired up with the corresponding item in String Parameter Values.

String Parameter Values

Optional. List of string values passed as parameters to the XSLT style-sheet. Each item is paired with the corresponding item in String Parameter Names.

String Parameter Namespace URI

Optional. Namespace URI of the parameters passed to the XSLT style-sheet.

Node Parameter List

Optional. List of nodes to pass as parameters to the XSLT style-sheet. The nodes in the list are paired with the corresponding item in the Node Parameter Namespace URIs argument.

Node Parameter Namespace URIs

Optional. List of namespaces that identify the corresponding object from the Node Parameter List argument in the style sheet.

Start Mode

Optional. Start mode for the XSLT transformation. This allows to specify a subset out of a larger XSLT style-sheet.

Start Mode Namespace URI

Optional. Namespace URI for the start mode. Allows to fully qualify the mode name.

Output

Result

String representation of the transformation result.

Exit Paths

Success

This path is taken if the transformation completed successfully.

Failure

This path is taken if the operation fails.

XML Unescape Entities

This XML tool returns a string with all occurrences of XML default entity references and character references replaced with the corresponding character. Resolved entities: `<`, `>`, `&`, `"`, `'`, `&#<decnum>`, `&#x<hexnum>`;

Inputs

Source String

String to process.

Normalize LF to CR/LF

Checkbox, default = False. Replace newlines (`\n`) with carriage-return/newline pairs (`\r\n`). Already present CR/LF pairs will not be expanded (i.e. CR/LF will not be expanded to CR/CR/LF)

Normalize CR/LF to LF

Checkbox, default = False. Replace carriage-return/newline pairs (`\r\n`) with a linefeed (`\n`).

Outputs

Result String

String with all entity and character references replaced by corresponding character.

Exit Paths

Next

This path always takes the Next exit path.

XML Validate Document

This XML tool performs run-time validation on the currently loaded document using the currently loaded DTD, schema, or schema cache.

Note: If the node is not a document node, the owner document of the node is validated.

Inputs

Document

The node of the document to be validated.

Exit Paths

Valid

Document successfully validated against schema or DTD.

Invalid

Document not successfully validated. Retrieve error information by calling [XML Get Error Info](#) on the Document node.

Failure

Other error (such as NULL node). Retrieve Error information through the XML Get Error Info on the Document node.

Initiators

Introduction to Initiators

An initiator is always the first step in a handler. It tells Interaction Processor which event starts an instance of that handler. When one of the modules in CIC, such as Telephony Services, generates an event, that event is seen by the Notifier. Notifier then tells other modules about that event. One of these modules is Interaction Processor, where the handlers are registered. When the Notifier tells the Interaction Processor about an event, Interaction Processor starts an instance of a handler.

When you publish a handler, the handler's initiator tells Interaction Processor which event to watch for. An event is something that happens to an object. For example, a *call* (object) can be *sent to voice mail* (event). If you configure an initiator in a handler to start when calls are sent to voice mail, then Interaction Processor starts that handler any time it is notified of that event.

Note: Subroutine initiators are different from other initiators because they are started by a call from another handler instead of an event that occurs on the CIC system. See the [Subroutine initiator](#) documentation for more information.

Initiator Properties

On the Initiator Properties page for most initiators, there are three settings. (HTML, Chat, Timer, and Subroutine initiators are different.) The combination of these three settings determine when Interaction Processor will start an instance of a handler containing that initiator. These three settings are described below:

- Notification Object Type

An object type is an entity that can be manipulated or managed by CIC. Examples of some of these objects include calls, chat sessions, and faxes. Queues are also objects, including User, Station, Workgroup, and line queues. The Notification Object Type of an initiator cannot be changed.

- Object ID

One property for an instance of an object is a unique object id. This is a unique number that differentiates one instance of an object from another. In most cases you should leave this setting to *any* so that the handler starts for an object with any Object ID. For

example, every call that comes through CIC has a unique Object ID. If you want a specific handler to run each time a specific event occurs to that call, set the Object ID to *any*. The default value for this setting is any to allow an initiator to start with all occurrences of an event to an object. For some initiators, you cannot change this setting.

- Notification Event

An event is an action that can occur to an object. Examples of some of the events that occur on a call object are transfer, send to voice mail, and new outgoing call request. Events also carry information that can be retrieved by the initiator for use in the handler. For example, caller ID is passed along as information in the event. The initiator can retrieve this information, assign it as the value of a variable, and then use the value from that variable in the handler.

The best way to understand how these three settings work together is to look at an example. For this example, pretend there is an initiator called Dog Barking. To understand how the Initiator Properties interact, look at the following examples:

Notification Object Type: **Dog**

Object ID: *{all}*

Event: **Bark**

With these settings, the initiator would start any time any dog barked.

Notification Object Type: **Dog**

Object ID: *Fido*

Event: **Bark**

With these settings, the initiator would start any time a dog named Fido barked.

Notification Object Type: **Dog**

Object ID: **Fido**

Event: *{all}*

With these settings, the initiator would start any time a dog named Fido did anything.

Notification Object Type: **Dog**

Object ID: *{all}*

Event: *{all}*

With these settings, the initiator would start any time any dog did anything.

Outputs

Initiators can pass information (attributes of the object) to a handler. The information initiators pass into handlers is specified on the initiator's Outputs page. For, example, the Incoming Call initiator passes into its handler the Call Identifier, the Line Identifier, the number the caller dialed, and other information. This information is used by a handler to perform actions on the incoming call object.

Initiators

[ACD Agent Available](#)

[ACD Call Timeout](#)

[ACD Process Call](#)

[Call Monitor](#)

[Call to Non-System Queue](#)

[Client Button Press](#)

[Client Prompt Request](#)

[Compile Voicemail TUI](#)

[Confirm Station Connection](#)

[Continuous Monitor Request](#)
[Custom Notification](#)
[Directory Services Change Notification Monitor](#)
[Email Interaction Disconnected](#)
[Email Interaction Incoming](#)
[Email Interaction Outgoing](#)
[Email Interaction Transferred](#)
[External Handler Call](#)
[Fax Send Completed](#)
[Generic Object In Non System Queue](#)
[Generic Object Monitor](#)
[GetDigitsExAsync](#)
[Held Interaction Timer](#)
[HTML Event](#)
[IC Change Notification Monitor](#)
[Incoming Call](#)
[Incoming Fax](#)
[Incoming Mail](#)
[Incoming SMS](#)
[Incoming Status Report](#)
[Incoming VPIM](#)
[Interaction Administrator Change Notification Monitor](#)
[Interaction Disconnected](#)
[Interaction Retry Later Attempt initiator](#)
[Interaction Snooze Timed Out initiator](#)
[Interaction Transferred to Queue](#)
[Keyword Spotted](#)
[Manage Survey Prompt](#)
[Message Light Notification](#)
[Messaging Request](#)
[Multi-Site Message Received](#)
[New Incoming Interaction](#)
[New Incoming Web Session](#)
[Object Disconnect](#)
[Outgoing Call Request](#)
[Outgoing Fax](#)
[Outgoing SMS](#)
[Parallel Make Call Outbound Call](#)
[Play Station Audio Request](#)
[Process Automation](#)
[Provision Station](#)
[Queue Period Statistics Report](#)
[Receive Log Events](#)

[Receive Traps](#)
[Run Survey](#)
[Secure Input](#)
[Send to Voice Mail](#)
[SMS in Non System Queue](#)
[SMS Monitor](#)
[SOAP Request](#)
[Station Off Hook](#)
[Subroutine](#)
[Switchhook Flash](#)
[Switchover Event](#)
[System Initialization](#)
[T1/E1 Wink Event](#)
[TCP/IP Connection Accepted](#)
[Timer](#)
[Transfer Request](#)
[Transfer to System Queue](#)
[UMF Message Received](#)
[User Status Monitor](#)
[Work Item Transferred to Queue](#)
[Wrapup Required](#)

ACD Agent Available Initiator

This initiator starts any time an AcdAgentAvailable event occurs. This event occurs any time an agent (a CIC client user who is a member of a workgroup designated to receive ACD interactions) logs into CIC or when an agent's status changes to available. See *CIC ACD Processing Technical Reference* located in the PureConnect Documentation Library for more information on ACD processing.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

ACD

Object ID

Select {all} for this initiator to work with all ACD objects.

Notification Event

Select AcdAgentAvailable.

Outputs

Agent Name

This is the name of the available agent.

Exit Paths

Start

This step always takes the Start exit path.

ACD Call Timeout Initiator

This initiator starts any time a call flagged for ACD processing waits too long in an ACD queue. You determine the in-queue time limit in the [ACD Process Call](#) initiator. Use this initiator to start a handler that performs special processing for items that have been waiting in a queue too long. One type of processing would be to place the call on an ACD queue with more agents or, if you are using skills, to make the required skills less restrictive. No further ACD processing will be done on this call until the ACD Process Call or ACD Process Chat tool is invoked. See the *CIC ACD Processing Technical Reference located in the PureConnect Documentation Library* for more information on ACD processing.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

ACD

Object ID

Select {all}.

Notification Event

Select AcdQueueItemTimeout.

Outputs

Call Identifier

The ID for the telephone call or chat session that timed out.

Workgroup Queue Name

The name of the workgroup where the call resided before it timed out.

Exit Paths

Start

This step always takes the Start exit path.

ACD Process Call Initiator

This initiator starts any time an AcdProcessQueueItem event occurs. This event is generated by the [ACD Initiate Processing](#) tool. A queue item is either a telephone call or chat session to be processed by ACD. See the *CIC ACD Processing Technical Reference* located in the PureConnect Documentation Library for more information on ACD processing.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

ACD

Object ID

Select {all}.

Notification Event

Select AcdProcessQueueItem.

Outputs

Call Identifier

The identifier for the call to be ACD processed.

Workgroup Queue Name

The name of the workgroup queue this call was in when the [ACD Initiate Processing](#) tool was called.

Exit Paths

Start

This step always takes the Start exit path.

Call Monitor Initiator

This initiator gathers information about a call object after an event occurs on that object. Many events can be selected in the Notification Event parameter. Some attribute values may be null, depending on where the call is when the event occurs. If the call has not been on the system for a long time, some attributes may not be present.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Call

Object ID

Select {all}.

Notification Event

Choose the type of event you want this initiator to start for. If you choose an event that is already in use in another Call Monitor Initiator, you may receive an error. This is because you might have two handlers trying to act on the same call object simultaneously.

Outputs

Call Identifier

The unique identifier for a call.

Line Type

Telephone lines from your local phone service provider are either analog lines (i.e., Plain Old Telephone - POTS - lines) or digital lines (e.g., T-1, ISDN, etc.).

Line ID

The unique identifier for a line.

Station Type

See the Interaction Administrator for a list of possible Station Types.

Station ID

The unique identifier for a station.

Local User ID

The User ID of the CIC client user participating in the call.

Local T Number

The telephone number of the CIC client user participating in the call.

Local Name

The name associated with the User ID of the CIC client user participating in the call.

Remote T Number

The unformatted telephone number of the person outside CIC who is making or receiving a call.

Remote Name

The name (if available) of the person outside CIC who is making or receiving a call.

Queue Type

Station, User, Workgroup.

Queue ID

The unique identifier for a queue.

Call Type

External: A non-CIC participant in a call.

Intercom: An CIC participant in a call.

External Party: A non-CIC conference call participant.

Intercom Party: An CIC conference call participant.

Call State

This is the current state of the call when the event occurred. For more information on call states see [States](#).

Call Total Hold Duration

The total time this call was on hold.

Initiated Date Time

The time this call was created.

Call Connect to Remote Date Time

The date and time the call became connected.

Call Terminated Date Time

The time the call object was terminated.

Call Direction

Inbound or outbound.

ACD Workgroup

If this was an ACD call, this is the name of the workgroup on which the call resided when the event occurred.

Number of Parties in Party Call

If this was a conference call, this is the number of callers participating in the conference call.

List of Call Ids for Parties in Party Call

The call ids for any calls in the conference call.

T1/E1 ANI/DNIS String

This is a value received by TS on T1/E1 lines for the ANI/DNIS string. It is raw, and un-decoded.

Remote Disconnect Flag

Flag used to indicate if the disconnect of the call was remote. It will be false if the disconnect was local, true if the disconnect was remote.

Exit Paths

Start

This step always takes the Start exit path.

Call to Non-System Queue Initiator

The purpose of the Call to Non-System Queue initiator is to start the handler System_CallOfferingNonSystemQueue and pass along information to that handler. A non-system queue is any queue other than the System queue including a user queue, a station queue, or a workgroup queue.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Call

Object ID

Select {all}.

Notification Event

Select Call to Non System Queue.

Outputs

Call Identifier

The identifier for the object that started this handler. A Call Identifier is a value that stays the same throughout the life of a call.

Line Identifier

The identifier for the line that this call object came in on. The contents of this variable corresponds to one of the lines set up in Interaction Administrator. Line identifier could be null if the call is an intercom call.

Workstation Identifier

The identifier for the intended recipient of this call.

Called Party Number

The identifier for the number the caller dialed.

Note: For an analog phone, there is only one called party number per line. Other types of lines may have multiple telephone numbers per line.

Calling Party Number

If available, this identifier contains the number of the calling party.

Calling Party Name

If available, this identifier contains the name of the calling party.

Queue Identifier

The name of the queue on which this call has been placed. This could be a user queue, a station queue, or a workgroup queue.

Call Initiation Time

The time the call was received for external calls. The time the call was placed for internal calls.

Exit Paths**Start**

This step always exits through the start path.

Client Button Press Initiator

This initiator fires when the client presses a custom toolbar button that was configured in Interaction Administrator with the "fire a handler" action.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Client Button

Object ID

Choose {all}.

Notification Event

Button pressed.

Outputs

Button Identifier

Key name assigned to the button.

Interaction Identifier

The unique identifier for the interaction.

User Name

Name of the user who triggered the event.

Station Name

Name of the station that triggered the event.

Exit Paths

Start

This step always exits through the start path.

Client Prompt Request initiator

This is the initiator that fires when a user records a personal prompt from a CIC client or over the telephone (through the remote voicemail retrieval handlers).

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Station Queue

Object ID

Select {all}.

Notification Event

Select Client Record Prompt Request.

Outputs

Station Queue Identifier

The unique identifier for the station from which the request to record a personal prompt was made.

User Queue Identifier

The unique identifier for the user queue associated with the station from which the request to record a personal prompt was made.

Prompt Type

The type of the recording to be made.

Note: An option 5 is available, which indicates that the recording is being made for Interaction Attendant.

Prompt Tag

The name of the recording. This is used by Interaction Attendant and contains the name specified by the Interaction Attendant user when recording a new audio file.

Exit Paths

Start

This step always exits through the Start path.

Compile Voicemail TUI Initiator

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Voicemail TUI Object Identifier

Object ID

All

Notification Event

All

Outputs

XML Filename

Exit Paths

Start

This step always exits through the start path.

Complete Async Receive Text from an Interaction Initiator

This initiator is triggered by the [Receive Text Async tool](#).

Outputs

Interaction Identifier

The ID of the interaction that received the Receive Text Async operation.

Context Value sent from Async Receive Text Request Tool

An arbitrary string set in the Receive Text Async tool to define the source of the operation (for example, the name of a handler).

Starting DateTime

The date and time the Receive Text Async tool started.

Timeout Set for Receive Text

The timeout value set in the Receive Text Async tool, in milliseconds.

Return value from Async Get Digit

0=Success

1=Failure

2=Timeout

3=Escape

Text

Text is returned if the initiator succeeds.

Exit Paths

Success

This path is taken if the text is successfully received.

Failure

This path is taken if the text is not successfully received.

Confirm Station Connection

This initiator will only fire when the telephony parameter "Confirm Station Connection" in Interaction Administrator is enabled. Otherwise, TsServer will follow its normal remote-station-connection behavior. For more information about the parameter, see the Interaction Administrator online help.

If the "Confirm Station Connection" parameter is enabled, TS will not immediately proceed with normal operation (placing the outbound call or connecting the picked up call) after the connection call is completed. Instead, it fires a Notification which triggers this initiator.

Once this initiator has fired, TS will then wait for one of the following things to occur:

1. A positive confirmation via the [Station Connection Confirmation](#) tool. If this occurs, TS will continue with the previous behavior.
2. A negative confirmation, in which case TS disconnects the connection call. This should cause a pending outbound call to disconnect and never be dialed, or fail the pickup on an inbound call.
3. Timeout. If this occurs, TS will continue as though a positive confirmation has been returned.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Call

Object ID

Select {all} for this initiator to work with all call objects.

Notification Event

Select "Confirm Station Connection Event".

Outputs

Call Identifier

The unique identifier of the call waiting for the connection.

Call Identifier

The unique identifier for the call being connected to.

Exit Paths

Start

This step always takes the Start exit path.

Continuous Monitor Request

A handler with this initiator will begin whenever a Continuous Monitor Request event is generated in the system.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Station Queues

Object ID

Select {all}.

Notification Event

ContinuousMonitorRequest

Outputs

Station Identifier

The unique identifier for the station making the continuous monitor request.

User Identifier

The unique identifier for the user making the continuous monitor request.

List of Queue Identifiers

A list string variable containing the identifiers of the queues to be monitored.

Supervisory Listen Flag

Set to indicate whether or not this request was made by a supervisor.

Exit Paths

Start

This step always takes the Start exit path.

Custom Notification Initiator

This initiator starts when a custom event occurs. You must specify the custom object id and the notification event that starts this initiator. One example might be a machine that reads credit card swipes. The machine that reads the credit cards has a unique object id. Any time a card is swiped, the Notification Event is card swipe. Therefore, when a card is swiped this initiator starts if the Object ID is card reader and the notification is card swipe.

Use this initiator to interface third-party products and applications with CIC.

There are two ways to create this event. You can use the [Send Custom Notification](#) tool, or you can use the SendCustomNotification.exe application that shipped with CIC. Both methods generate the event that starts this initiator.

Initiator Properties Page

Notification Object Type

Custom is your only choice for this parameter.

Object ID

Type the Object ID passed in by the Send Custom Notification tool or SendCustomNotification.exe application.

Notification Event

Type the Notification Event passed in by the Send Custom Notification tool or SendCustomNotification.exe application.

Outputs

Custom Object Identifier

The identifier for the custom object.

Custom Event Identifier

The name of the event that started this initiator. If you set Notification Event to All, you can use the information in this string to figure out which event started this handler.

Custom Notification Data

Any information passed along with the Notification Event. Remembering the example of the card reader, this might be the information on the card. This information is passed as a list of strings.

Exit Paths

Start

This step always exits through the start path.

Directory Services Change Notification Monitor Initiator

This initiator starts when Directory Services (DS) makes one or more changes to an attribute within a key. This information is used for reporting changes made to DS. This initiator is similar to the [IC Change Notification Monitor initiator](#).

Note: In general, if you must monitor stations (or users and workgroups) for configuration changes, we recommend using the IC Client COM API as the most efficient and safest way to monitor changes. If you do not have Client COM, we recommend IC Change Notification Monitor initiator rather than the Directory Services Change Notification monitor as the next best approach. You must test all changes carefully when using these Monitor initiators to make sure high level access level or security changes don't cause excessive notifications.

For example, when a user's security privileges are changed, DS writes that change to an attribute within that user's User key. DS then generates a notification that it has made the change. The Directory Services Change Notification Monitor initiator then starts and retrieves information about that change from the notification. A notification is generated for each key that changes, and the attributes that changed within that key are listed in a list of string type variables.

Note: This initiator is not currently used in the handlers that ship with CIC.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

IC Change Notification

Object ID

This should always remain as {all}.

Notification Event

This specifies the type of change event to monitor. If you want to monitor for all four types of changes, you need to use different handlers that call a common subroutine.

Created Active - fires when an active DS key is created

Deleted Active - fires when an active DS key is deleted

Renamed Active - fires when an active DS key is renamed

Modified Active - fires when an active DS key is modified

Note: All DS keys under the root and site name are considered "active" and subject to monitoring. The root DS key path is: HKEY_LOCAL_MACHINE\SOFTWARE\Interactive Intelligence\CIC\Directory Services\Root. Any keys outside of this tree are considered "inactive."

Outputs

You have the option to create a custom variable to hold each of these outputs, if necessary. If you do create custom variables, you will need to modify the rest of the handlers or subroutines appropriately instead of relying on the default variables provided in this initiator. To create a custom variable, select the **(new)** option at the top of each drop-down list and specify the variable name on the Declare Variable dialog.

Entry Path

The full path to the DS Key, including the name of the Key, starting in the DS root directory.

Entry Name

The name of the key in which the change was made.

Object Class

Class tells you what kind of object was changed (User, Line, Line Group, Handler, etc). For example, if the change takes place to a specific user's configuration, the Object Class would be User. If someone changed a report log, then the Object Class would be

Report Log.

Notification Type

The type of change that took place is described by one of the following values:

- "Created"
- "Deleted"
- "Renamed"
- "Modified"

List Changes Attributes

A list of the attributes that were changed within the Key.

Exit Paths

Start

This step always exits through the start path.

Email Interaction Disconnected Initiator

This initiator starts when a user disconnects an email interaction.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Label (?)

Email Interaction Disconnected.

Notification Object Type

Email Interaction

Object ID

Select {all}.

Notification Event

Email Interaction Disconnected.

Outputs

Interaction Id

The identifier for the interaction to disconnect.

Exit Paths

Start

This step always takes the Start exit path.

Email Interaction Incoming Initiator

This initiator when a new email interaction comes into the system.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Label (?)

Email Interaction Incoming

Notification Object Type

Email Interaction

Object ID

Select {all}.

Notification Event

Email Interaction Incoming

Outputs

Interaction Id

The identifier for the incoming interaction.

Queue

The queue for the incoming email interaction.

Reserved

Reserved for future use.

Exit Paths

Start

This step always takes the Start exit path.

Email Interaction Outgoing

This initiator starts when a user sends an email interaction from the client.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Label (?)

Email Interaction Outgoing

Notification Object Type

Email Interaction

Object ID

Select {all}.

Notification Event

Email Interaction Incoming

Outputs

Interaction Id

The identifier for the incoming interaction.

Queue

The destination queue for the outgoing email interaction.

Reserved

Reserved for future use.

Exit Paths

Start

This step always takes the Start exit path.

Email Interaction Transferred Initiator

This initiator starts when a user transfers an email interaction to another queue.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Label (?)

Email Interaction Transferred

Notification Object Type

Email Interaction

Object ID

Select {all}.

Notification Event

Email Interaction Transferred

Outputs

Interaction Id

The identifier for the transferred email interaction.

Queue

The destination queue for the transferred email interaction.

Parked

A Boolean (true/false) value to indicate if the email interaction is parked on the destination queue.

Exit Paths

Start

This initiator always takes the Start exit path.

External Handler Call Initiator

This initiator allows a handler to be initiated via an inline C++ function call. This inline function triggers the ExternalHandlerCall initiator and gathers the result of the ExternalHandlerReturn tool.

The function call contains the following arguments:

First argument: a string for the operation name. The operation name is the object ID of the notification event that will trigger the initiator.

Second argument: a list of strings that will be passed to the external handler call initiator.

Third argument: the timeout period in milliseconds for the function call to return.

Fourth argument: a list of strings that are passed back from the external handler return tool.

Notification Object Type

External Handler Call

Object ID

The object ID for this initiator is the operation name passed as the first input argument to the ExternalHandler C++ function call.

Event ID

Unique identifier reserved for internal use only.

Outputs

RequestHandle

A system generated identifier for this external handler call. The request handle allows this initiator to correlate with the External Handler Return tools.

Data

A list of strings corresponding to the second input argument to the ExternalHandler C++ function call.

Exit Paths

Start

This initiator always takes the Start exit path.

Fax Send Completed

A handler with this initiator runs any time a fax has been sent successfully, or if the fax was not sent for some reason. After a handler with this initiator runs, the fax is deleted from the fax server.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Fax Envelope

Object ID

Select {all}.

Notification Event

FaxSendComplete

Outputs

Fax Envelope ID

The unique identifier for the fax that was sent.

Fax ID

The unique identifier for a fax.

Call Identifier

The call id for the call the fax went out on.

Send Successful

This value is true if the fax was sent successfully.

Recipient Fax Number

The fax recipient's phone number.

Recipient Company Name

The fax recipient's company name.

Recipient Name

The fax recipient's name.

Sender Name

The sender's name.

Sender's Fax Number

The sender's fax number.

Sender's Telephone Number

The sender's telephone number.

Sender's Company Name

The sender's company name.

Notify Sender when fax is sent

True if you want the sender to be notified if the fax was sent successfully; otherwise, false.

Email address to notify if fax when fax is sent

The sender's email address.

Notify Sender if fax fails

True if you want to sender to be notified if the fax was not sent successfully; otherwise, false

Email address to notify if fax cannot be sent

The sender's email address, or some other address to notify if the fax was not sent successfully.

FaxDevice

The name of the fax device.

Fax transmission rate

The actual transmission rate at which the fax was sent.

Send Type

The category of time when the fax should be sent.

Scheduled Time

The time the fax should be sent if the Send type is one.

Time of day low rates begin

The begin time a fax is sent if the Send Type was two.

Time of day low rates end

The latest time a fax is sent if the Send Type was two.

Login name user or handler that created this fax

The network user id of the person sending the fax.

Failed Attempts

The number of failed calls for that fax. For example, if the sender had configured their fax for 3 retries, the Fax Server might try to send it up to four times. Depending on why the try failed, Fax Server may or may not attempt another retry. For reasons such as busy signal and no answer, retries are attempted (if configured by the sender). For other cases such as operator intercept tones, that entire fax job fails and no more retries are attempted.

Reason Call Failed

The reason the call failed.

Call Duration

How long the call lasted.

Remote Calling Station Identifier

The sending fax station's line number or other unique identification.

Number of Pages Sent/Returned

The total number of pages sent or returned.

History of all Fax Attempts

This string contains the complete history of the fax's activities, starting when the fax is queued for sending, and ending when the fax is successfully sent or the max number of attempts is met.

Exit Paths

Start

This step always exits through the start path.

Generic Object In Non System Queue Initiator

Starts whenever a Generic Object is sent to a non-system queue. A non-system queue is any queue other than the system queue, including user and workgroup queues.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Generic Object

Object ID

Select {all}.

Notification Event

Select Generic Object in Non-System Queue.

Outputs

Generic Object ID

The identifier for the generic object that entered a non-system queue.

Queue Name

The queue the generic object Question has entered.

Exit Paths

Start

This step always takes the Start exit path.

Generic Object Monitor Initiator

This initiator gathers information about a generic object after an event occurs on that object. Many events can be selected in the Notification Event parameter. Some attribute values may be null, depending on where the email is when the event occurs. If the object has not been on the system for a long enough time, some attributes may not be present.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Generic Object

Object ID

Select {all}.

Notification Event

Choose the type of event for which you want this initiator to start.

Outputs

Generic Object ID

The unique identifier for the generic object.

Exit Paths

Next

This step always takes the Next exit path.

Get Digits Ex Async Initiator

This initiator fires in response to a caller pressing one or more keys after an [Extended Get Key Async](#) tool has been called by a handler. Use of this imitator in conjunction with the Extended Get Key Async tool allows ACD calls to be placed on hold without causing the processing handler to remain paused. The handler that was initially processing the call will continue after the call is placed on hold, and a new handler will be called via this initiator. No further processing will be done on the call by the first handler after the Extended Get Key Async tool has exited.

Note: only one handler my use this initiator on any given system.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Call

Object ID

Select {all}.

Notification Event

Select GetDigitsExAsync.

Outputs

Call Identifier

The unique identifier for the call.

Correlation ID

Specifies where the Extended Get Key Async tool that prompted this initiator came from. Typically this will be the ACDAudioOnHold handler.

Termination Code

The reason the Get Key operation ended. Valid termination codes are:

Code Meaning

- 0 Success
- 1 Timeout
- 2 Attribute Changed
- 3 Escape
- 4 Tone
- 5 Failure
- 6 No Response

Keys

The keys retrieved by the Extended Get Key Async tool.

Exit Paths

Next

This step always takes the Next exit path.

Held Interaction Timer Initiator

This initiator starts when a interaction remains on hold longer than the number of seconds configured for the Held Interaction Timeout server parameter. This server parameter is defined in Interaction Administrator and defaults to 900 seconds. A timer starts each time a call is put in a held state. This initiator replaces the Held Call Timer Initiator.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Call

Object ID

Select {all}.

Notification Event

HeldTimerExpired.

Outputs

Interaction Identifier

The interaction ID for the interaction that remained on hold too long.

Interaction was parked?

The initiator only fires for interactions that are on hold. It does not fire for parked interactions.

Exit Paths

Start

This initiator always takes the Start exit path.

HTML Event Initiator

The HTML Event initiator begins when an HTML event is generated. This might occur when a person interacting with a web page fills out a web page form and clicks the submit button. See the *Interaction Web Tools Developer's Guide in the PureConnect Documentation Library*.

Event Page (HTML Event Initiator)

Registered Name

The name of this initiator which is registered on the CIC server when this handler is published. The event and field information is stored along with the initiator's name on the CIC server and this allows this information to be recalled in another HTML initiator by selecting the proper registered name. Inserting the Registered Name of a previously used HTML Event initiator will set all the settings to be those of the named HTML Event initiator.

Example: HandlerA is written using the HTML Event initiator and the Registered Name is specified as "HandlerA." This stores all the settings specified for HandlerA under that name on the CIC server. Later, you're writing a second handler, HandlerB, that uses all

the same settings as HandlerA. Instead of manually assigning them each individually, simply insert "HandlerA" into the Registered Name field of HandlerB, and all the settings will be automatically set to those saved under "HandlerA."

HTML Event

The first command line argument to the CIC servlet.

Field List

The form element names passed in from the servlet.

Variable List

The variables in the handler that contains the values of the form element names.

Add button

Click this button to add a form element and an associated variable.

Edit button

Click this button to edit a form element and its associated element.

Delete button

Click this button to delete a selected form element.

CGI Command Line Arguments

The variable of type List of String that holds any command line arguments passed along with the event.

Web Connection

The variable that contains the name of the web connection. The web connection is established by the CIC servlet. The web connection is used by the Generate HTML step to send a web page back to a user. If there is no Generate HTML step to send a page back to a user, the user is sent a page that says "Error Generating HTML."

Outputs

These outputs are generic optional servlet variables. You will most likely never need to specify values for these parameters. These environment variables are set when the server executes the gateway program.

Gateway Interface

Contains the revision of the CGI specification that this server uses.

Request Method

The method used for the request. It tells you where and how to look for whatever data is passed. Usually it is either Post or Get.

Script Name

This is set to the file name of the servlet. This may be useful if you are generating scripts on the fly.

Query String

This variable contains information being passed to the servlet. If you are trying to debug your HTML query strings, you can store their values here.

Server Software

The name and version of the information server software answering the request (and running the gateway).

Server Name

The server's host name, DNS alias, or IP address as it would appear in self-referencing URLs.

Server Protocol

The name and revision of the information protocol accompanying a request.

Server Port

The port number to which the request was sent.

HTTP User Agent

Contains the name and version of the user's browser.

HTTP Accept

Provides the MIME format that the browser can accept.

Path Info

The extra path information, as given by the client. In other words, scripts can be accessed by their virtual path name, followed by extra information at the end of this path. The extra information is sent as PATH_INFO. This information should be decoded by the server if it comes from a URL before it is passed to the servlet.

Path Translated

The server provides a translated version of PATH_INFO, which takes the path and does any virtual-to-physical mapping to it.

Remote Host

Contains the text equivalent of the host name of the computer connected to your web site. Specify a value here if you want to log this information.

Remote Address

Contains the IP address in dotted-decimal notation of the computer making the request. Specify a value here if you want to log this information.

Remote User

The name of the person connected to your web site. This information is not present if the user has not specified a name in his or her browser.

Remote Identifier

If the HTTP server supports RFC 931 identification, then this variable is set to the remote user name retrieved from the server. Use of this variable should be limited to logging only.

Authentication Type

Contains the authentication method used to validate the user.

Content Type

The MIME content-type of the attached information.

Content Length

The number of bytes of data in the attached data.

HTTP Cookie

The information contained in the HTTP cookie of the user connected to your web site. The information is stored as a '&'-separated set of values. An example would be:

Brian & Depauw Blvd.&Suite 1060&&Indianapolis&IN&46268&ExistingCustomer&Marketing

The subroutine SystemWebParseCookie will break apart this stream of data into the appropriate data containers(variables).

Exit Paths

Start

This step always exits through the start path.

IC Change Notification Monitor Initiator

This initiator starts when Admin Server makes one or more changes to an attribute within a key. This information is used for reporting changes made to Admin Server (e.g., via Interaction Administrator, a CIC client, etc.). It is similar to the [Directory Services Change Notification Monitor initiator](#), but the IC Change Notification Monitor is more efficient as it only fires when changes are made on a specific class.

Note: In general, if you must monitor stations (or users and workgroups) for configuration changes, we recommend using the IC Client COM API as the most efficient and safest way to monitor changes. If you do not have Client COM, we recommend IC Change Notification Monitor initiator rather than the Directory Services Change Notification monitor as the next best approach. You must test all changes carefully when using these Monitor initiators to make sure high level access level or security changes don't cause excessive notifications.

Admin Server keeps a cached copy of Directory Service (DS) information. For example, when a user's security privileges are changed, DS writes that change to an attribute within that user's User key. DS then generates a notification that it has made the change. Admin Server makes the change to its cached DS information. The IC Change Notification Monitor initiator then starts and retrieves information about that change from the notification. A notification is generated for each key that changes, and the attributes that changed within that key are listed in a list of string type variable.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

IC Change Notification

Object ID

{all}

Notification Event

{all}, or select one of the event types from the list. The events listed are for changes to user and workgroups.

Outputs

Entry Path

The full path to the DS Key, including the name of the Key, starting in the DS Root directory.

Entry Name

The name of the key in which the change was made.

Object Class

Class tells you what kind of object was changed. (User, Line, Line Group, Handler, etc). For example, if the change takes place to a specific user's configuration, the Object Class would be User. If someone changed a report log, then the Object Class would be Report Log.

Notification Type

The type of change that took place is described by one of the following values:

- "Created"
- "Deleted"
- "Renamed"
- "Modified"
- "Inactive Created" (if the Configuration Set is marked inactive in DS)
- "Inactive Deleted" (if the Configuration Set is marked inactive in DS)
- "Inactive Renamed" (if the Configuration Set is marked inactive in DS)
- "Inactive Modified" (if the Configuration Set is marked inactive in DS)

List Changes Attributes

A list of the attributes that were changed within the Key.

Exit Paths

Start

This step always exits through the start path.

Incoming Call Initiator

This initiator starts when a call comes into the system. If available, ANI and DNIS (caller id information) and other information about that call is passed into the handler.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Call

Object ID

Select {all}

Notification Event

Select NewIncomingEvent.

Outputs

Call Identifier

The unique identifier for the incoming call.

Line Identifier

The unique identifier for the line on which the call is coming in.

Called Party Number

The number dialed by the caller.

Calling Party Number

The caller's telephone number.

Calling Party Name

The caller's name, if available.

Exit Paths

Start

This step always exits through the start path.

Incoming Fax Initiator

This initiator starts any time a new incoming fax comes into the system.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Fax Envelope

Object ID

Select {all}.

Notification Event

Select Incoming Fax.

Outputs

Fax Envelope ID

The unique identifier for a Fax Envelope.

Fax ID

The unique identifier for a fax.

Call Identifier

The unique identifier for a call a fax comes in on. Call Id can be useful for obtaining information about the call. You can route a fax based on who a fax comes from, or route based on some other attribute.

T.30 Subaddress received with fax

A handshake protocol used by the machine sending the fax. This information can be used to identify who sent a fax. See the documentation that accompanied your fax hardware for more information.

Fax device on which this fax was received

The identifier for the fax station (stand-alone fax, SCBus fax, etc) that received the fax. This value is one of the fax stations set up in Interaction Administrator.

Remote Calling Station Identifier

The ID of the Remote Calling station. This information is sent by the remote fax. It can be useful when a fax is forwarded in email.

Call Duration

The length of the fax transmission in seconds.

Fax Transmission Rate

The transmission rate at which the fax was received.

Exit Paths

Start

This step always exits through the start path.

Incoming Mail Initiator

Handlers with this initiator are run when mail arrives in a mailbox (e.g., Exchange Inbox, IBM Notes, Interaction Message Store, etc.) after it is configured as a Monitored Mailbox in Interaction Administrator. In Interaction Administrator, open the Mail configuration container (under System Configuration) and select the Monitored Mailboxes tab. From here, you create a monitored mailbox and specify the mailbox to monitor. See the Interaction Administrator online help for "Monitored Mailbox" for configuration details.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Mailbox

Object ID

The values available here are dependent upon the CIC server configuration. The drop-down list contains an identifier for each mailbox that the Interaction Processor is currently configured to monitor, and the "catch all" identifier {all}

Notification Event

The only mailbox event is "New Mail."

Outputs

Cookie

This initiator outputs an e-mail cookie type of variable that carries all of the e-mail and mailbox information that other tools need.

Incoming SMS

This initiator handles eSMSEvent_NewIncomingSMS events. This primary initiator is typically processed by the System_IncomingSMS handler.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

SMS Object

Object ID

Select {all}.

Notification Event

Incoming SMS

Outputs

SMS Identifier

The unique identifier for the SMS object.

Ticket Identifier

The SMS Broker's ticket ID for this SMS Object.

Queue Identifier

The identifier for the queue on which the SMS object exists.

Remote Name

The sender's name.

Remote Telephone

The sender's telephone number.

Local Location

The dialed number.

Short Message

The message of the SMS Object.

Local Alias

The alias used for collective phone numbers.

Local Account

The SMS account information.

Date Received

The date the message was received.

Date Delivered

The date the message was sent by the SMS Broker.

Date Broker

The date the message was sent to the SMS Broker.

Mobile Country Code

A number identifying the country the message was sent from.

Mobile Network Code

A number identifying the network used by the sending cell phone.

Exit Paths

Next

This step always takes the Next exit path.

Incoming Status Report

This primary initiator is typically processed by the System_IncomingSR handler to process status reports sent by an SMS Broker.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

SMS Object

Object ID

Select {all}.

Notification Event

Select Incoming Status Report.

Outputs

SMS Identifier

The unique identifier for the SMS Object.

Ticket Identifier

The SMS Broker's ticket ID for this SMS Object.

Message Status

The current status of the SMS Object.

Detailed status of the message

An explanation of the Message Status. For example, if the Message Status indicates that the message was not sent, this will explain why it could not be sent (like an invalid option, invalid phone number, etc).

Client ID of the message

The ID of the client that initiated the SMS Object.

Date Broker

The date the message was sent to the SMS Broker.

Exit Paths

Next

This step always takes the Next exit path.

Incoming VPIM

This initiator starts any time a new incoming VPIM comes into the system.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Incoming VPIM Message Params.

Object ID

Choose {all}.

Notification Event

Choose "Incoming VPIM."

Outputs

XML data containing information about the VPIM message data

String containing selected information about the VPIM message.

Exit Paths

Start

This step always exits through the start path.

Interaction Administrator Change Notification Monitor Initiator

This initiator starts when a user makes a configuration change. The information this initiator retrieves from the notification is used to report on changes users make from Interaction Administrator or other applications (for example, a user changes a call setting in Interaction Connect.).

Note: This initiator is not used in the handlers that are included with CIC.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Interaction Administrator Change Notification

Object ID

{all}

Notification Event

Select IA Change Event.

Outputs

Change Type

Always one of the following types of changes:

- Addition
- Modification
- Deletion

Time of Change

The date and time the change was made.

User

The User ID of the user who made the change.

Station

The Station ID of the computer from where the change was made.

Class

A value used internally to indicate what type of value was changed.

Key

The Directory Services key in which the change was made. For example, this might be the User Key, Workgroup Key, etc.

Application Name

The application used to make a change that caused the initiator to start. This output contains "Interaction Administrator" if Interaction Administrator was used to make a change and an empty string if any other application was used to make the change.

Exit Paths

Start

This step always exits through the start path.

Interaction Disconnected Initiator

A handler with this initiator will run whenever a web interaction is disconnected.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Interaction Object.

Object ID

Select {all}.

Notification Event

Interaction Disconnected Event.

Outputs

Interaction ID

The unique identifier of the disconnected interaction.

Exit Paths

Next

This step always takes the Next exit path.

Interaction Retry Later Attempt initiator

This initiator starts when an agent places a web interaction in a “Snooze” state. This initiator starts the System_RetryInteractionLater handler to call the Snooze Interaction toolstep to place the interaction in the “Snooze” state. When the “Snooze” period ends, the Interaction Snooze Timed Out initiator starts the System_RetryInteractionTimeout handler to place the interaction in the queue to start ACD processing.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Interaction Object.

Object ID

Select {all}.

Notification Event

Select the event to initiate this handler.

Outputs

Interaction Identifier

The unique identifier of the web interaction.

Custom Retry Data Names

A list of variable names to use for custom data.

Custom Retry Data Values

A list of values for each of the variables listed in the Custom Retry Data Names parameter.

Exit Paths

Start

This step always takes the Start exit path.

Interaction Snooze Timed Out initiator

This initiator starts when the “Snooze” period ends for a web interaction. When the “Snooze” period ends, this initiator starts the System_RetryInteractionTimeout handler to place the interaction in the queue to start ACD processing.

The Interaction Retry Later Attempt initiator starts the System_RetryInteractionLater handler to call the Snooze Interaction toolstep when an agent places the interaction in the “Snooze” state.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Interaction Object.

Object ID

Select {all}.

Notification Event

Select the event to initiate this handler.

Outputs

Interaction Identifier

The unique identifier of the web interaction.

Queue Identifier

The unique identifier for a queue.

Exit Paths

Start

This step always takes the Start exit path.

Interaction Transferred to Queue Initiator

This initiator is fired when a web interaction is transferred to a non-system queue. The System_InteractionOnNonSystemQueue handler is launched with this initiator.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Interaction Object

Object ID

Select {all}.

Notification Event

Interaction on Non System Queue.

Outputs

Interaction ID

The unique identifier for the interaction.

Queue Identifier

The queue to which the interaction was transferred.

Exit Paths

Next

This step always takes the Next exit path.

Keyword Spotted Initiator

This initiator is a way for handlers to subscribe to keyword spotted events. There should be one such event for each keyword spotted.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Label

Keyword Spotted

Notification Object Type

Call

Object ID

Select {all}.

Notification Event

Select {all}.

Outputs**Call Identifier**

The identifier for the call in which the keyword was spotted.

Keyword Name

Name of the keyword that was spotted.

Keyword Set ID

The ID of the keyword set to which the keyword belongs.

Utterance

Select Utterance.

Absolute Start Time

The absolute time stamp when the spotted keyword started, as reported by the media server.

Duration (ms)

The duration, in milliseconds, of the audio for the spotted keyword.

Confidence

The confidence level as reported by the media server. This is a fractional number between 0.0 and 1.0 that indicates the level of confidence that the spoken word is actually the word that is Interaction Analyzer is trying to detect.

Channel

Identifies the person who spoke (either the agent or the customer).

Keyword Tag

A tag that Interaction Administrator assigns to an Interaction Analyzer keyword.

Keyword Set Tag

A tag that Interaction Administrator assigns to the Interaction Analyzer keyword set to which the keyword belongs.

Keyword Set Category

A category that Interaction Administrator assigns to the Interaction Analyzer keyword set for the spotted keyword.

Exit Paths

Start

This initiator always takes the Start exit path.

Manage Survey Prompt Initiator

This initiator starts when the client sends a notifier event, after a user requests to play or record a prompt for a customer satisfaction survey.

Note: This initiator will not be functional until a future service update.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

CCSurvey Object

Object ID

Select {all}.

Notification Event

Select {all}.

Outputs

Prompt Recording SessionId

The session ID associated with the prompt recording.

Station Queue Id

The station queue ID for the interaction.

User Queue Id

The user queue ID for the interaction.

Phonenumber

The destination phone number.

Filename

The name of the file to play or to use for the recording.

Action: Play or Record

Indicates whether the user requested to play or record a prompt.

Message Light Notification initiator

This initiator starts when the MLUtil DLL is executed. (MLUtil DLL is currently used by the voice mail form to send message light notifications, starting this initiator.) It represents either a new message being added to a user's inbox, or a message being read or removed from a user's inbox.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Call

Object ID

Select {all}.

Notification Event

Select Message Light Notification.

Outputs

Operation

This integer value is 0 if the message was added to the user's inbox, 1 if the message was read or deleted from the user's inbox.

IC User Id

This string value is the CIC User ID of the user who has the message.

Number of unread Email messages

This integer value is the number of unread email messages in the user's inbox, or -1 if unknown. A -1 indicates that the process sending the notification could not determine these values, and that the handler should determine these values.

Number of unread Voice Mail messages

This integer value is the number of unread voice mail messages in the user's inbox, or -1 if unknown. A -1 indicates that the process sending the notification could not determine these values, and that the handler should determine these values.

Number of unread Fax Mail messages

This integer value is the number of unread fax mail messages in the user's inbox, or -1 if unknown. A -1 indicates that the process sending the notification could not determine these values, and that the handler should determine these values.

Exit Paths

Next

This step always takes the Next exit path.

Messaging Request Initiator

This initiator is fired when a web interaction is transferred to voicemail.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Interaction Object

Object ID

Select {all}.

Notification Event

Interaction To Voicemail Request

Outputs

Interaction ID

The unique identifier for the interaction.

Queue Identifier

The queue containing the interaction.

Interaction Transferred from Client

Use of this output parameter is deprecated.

Exit Paths

Next

This step always takes the Next exit path.

Multi-Site Message Received

This tool is the initiator for a handler that will send a specific message.

Initiator Properties

Label

The label that is displayed for this tool step on the main ID screen. The default label is "Multi-Site Message Received."

Object ID

A string value that matches the Object ID set in [Multi-Site Send Event](#) or [Multi-Set Send Request](#).

Notification Event

A string Value that matches the Object ID set in Multi-Site Send Event or Multi-Set Send Request.

Outputs

Message Handle

The handle of the incoming message from which the data elements are to be read. The message handle is a value that is required as input for all tools that operate on the message.

Response Correlation ID

A unique identifier for this message if the message was sent synchronously. Otherwise, this value will be zero. If the message requires a response, the value of the variable will contain a non-zero value. This value must be passed unchanged to the [Multi-Site Send Response](#) tool.

Response Destination ID

A value that identifies the source of the request if the message was sent synchronously. If the message was sent asynchronously, this value will be an empty string. This value must be passed unchanged to the Multi-Site Send Response tool.

Note

The values in Response Correlation ID and Response Server ID should not be changed by the handler.

New Incoming Interaction Initiator

This initiator will fire on all new incoming web interactions, such as chats, callbacks or web collaborations.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Interaction Object

Object ID

Select {all}.

Notification Event

New Incoming Interaction

Outputs

Interaction ID

The unique identifier for the incoming interaction.

Queue Identifier

The identifier for the queue containing the interaction.

Exit Paths

Next

This step always takes the Next exit path.

New Incoming Web Session Initiator

This initiator will fire on all new incoming web interactions, such as someone browsing to a tracked page, or requesting a callback or web collaboration.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Web Session Object

Object ID

Select {all}.

Notification Event

New Incoming Interaction

Outputs

Interaction ID

The unique identifier for the interaction.

Exit Paths

Next

This step always takes the Next exit path.

Object Disconnect Initiator

This initiator fires whenever an object is disconnected from a queue.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Queue Item

Object ID

Select {all}.

Notification Event

Select QueueItemDisconnected

Outputs

Call Identifier

The unique identifier for the interaction.

Exit Paths

Start

This step always takes the Start exit path.

Outgoing Call Request Initiator

This initiator begins when a person dials an outgoing number from a CIC Client.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Call

Object ID

Select {all}.

Notification Event

Select Outgoing Call Request.

Outputs

Call Identifier

This variable is the identifier for the object that started this handler. A Call Identifier is a value that stays the same throughout the life of a call.

Number being dialed

The identifier for the number the caller dialed. A comma causes a two-second pause, and any numbers after the "/" symbol are dialed after the call is connected.

Base number being dialed

The telephone number with any numbers after the "/" symbol removed. For example, if a user calls 1-317-872-3000/103, the base number would be 1-317-872-3000. The base number can be used for security lookups.

Line groups

The identifier for the line that this call object can go out on. The contents of this variable correspond to one of the lines set up in Interaction Administrator.

Station Queue Identifier

The Station queue on which this call originated.

User Queue Identifier

The name of the user queue on which this call originated.

Calling Party Number

This parameter passes a string of digits to be displayed as ANI or Caller ID on the phone system that receives the call. If you do not specify a value in this parameter, the Telephony Services subsystem uses the number entered in the Phone Number field in the Line Configuration container in Interaction Administrator.

Note: This only works for ISDN.

Calling Party Name

This parameter passes a string to display the name of the caller on the phone system that receives the call. If you do not specify a value in this parameter, the Telephony Services subsystem uses the name associated with the number entered in the Phone Number field in the Line Configuration container in Interaction Administrator.

Note: This only works for ISDN.

Exit Paths

Start

This step always exits through the start path.

Outgoing Fax Initiator

This initiator starts any time a fax is sent manually from Interaction Fax or the Interaction Fax print driver.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Fax Envelope

Object ID

Select {all}.

Notification Event

Outgoing Fax

Outputs

Fax Envelope ID

The unique identifier for the outgoing fax envelope. See the [Create Envelope](#) tool for more information on fax envelope.

Fax ID

The unique identifier for a fax.

Recipient Fax Number

The fax recipient's phone number.

Recipient Company Name

The fax recipient's company name.

Recipient Name

The fax recipient's name.

Sender Name

The sender's name.

Sender's Fax Number

The sender's fax telephone number.

Sender's Telephone Number

The sender's telephone number.

Sender's Company Name

The sender's company name.

Notify sender when fax is sent

Set this to true if you want the sender to be notified if the fax was sent successfully; otherwise, false.

Email address to notify if fax when fax is sent

The sender's email address.

Notify sender if fax fails

Set this to true if you want to sender to be notified if the fax was not sent successfully; otherwise, false.

Email address to notify if fax cannot be sent

The sender's email address, or some other address to notify if the fax was not sent successfully.

Cover Page Comments

Text contained in the comments field (if any) on the cover page.

Page Header Comments

Header information for the cover page.

Cover page Type

Only Interaction Fax cover page documents (with a file extension of .i3c) are valid in this release. This could change in future releases.

Cover Page Name

The name of the cover page document.

Fax Workstation Group Name

The station or fax station on which to send the fax.

Number of Retries to Attempt

How many times CIC will attempt to send the fax.

Delay between retries

How long to wait between retries.

Maximum BPS to attempt

The maximum BPS to attempt. Usually you should set this to 0.

Send Type

The category of time when the fax should be sent.

Scheduled Time

The time the fax should be sent if the Send type is one.

Time of day low rates begin

The begin time a fax should be sent if the Send Type was two.

Time of day low rates end

The latest time a fax should be sent if the Send Type was two.

Login name user or handler that created this fax

The network user id of the person or handler sending the fax.

Exit Paths

Start

This step always exits through the start path.

Outgoing SMS

This monitor initiator launches whenever a new, outgoing SMS Object is detected.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

SMS Object

Object ID

Select {all}.

Notification Event

Outgoing SMS.

Outputs

SMS Identifier

The unique Identifier for the SMS Object.

Exit Paths

Next

This step always takes the Next exit path.

Parallel Make Call Outbound Call Initiator

This initiator is fired by the [Parallel Make Call](#) tool.

Initiator Properties

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Call.

Object ID

Select {all}.

Notification Event

Select Parallel Make Call Outbound Call.

Outputs

Call Identifier

This variable is the identifier for the object that started this handler. A Call Identifier is a value that stays the same throughout the life of a call.

Telephone Number

The identifier for the number being dialed.

Flag to indicate this is an intercom call

Indicates that the call should be placed as an intercom call as opposed to an outbound call. When Parallel Make Call Dial is called, the value of this flag is passed to the intercom call input parameter.

Station Identifier

The identifier for the station called.

Workgroup

The identifier for the workgroup called.

Perform Call Analysis

Indicates whether or not Call Analysis is to be performed on this call.

Include Answering Machine Detection in Call Analysis?

Indicates whether or not Call Analysis is to check for an answering machine.

Silence Wait Timeout

The maximum number of seconds of silence to elapse after a call has been answered.

No Answer Timeout

The number of seconds to wait for a call to be answered.

Global Parameter String

The parameter that is sent to each instance of this initiator.

Local Parameter String

A string specific to this instance of the initiator. For more information, see the [Parallel Make Call](#) tool.

Exit Paths

Next

This initiator always takes the Next exit path.

Play Station Audio Request initiator

This initiator starts a handler that plays a .WAV file to a listener through a telephone handset. This initiator is reserved for use by the Interaction Recorder application to play recordings to Interaction Recorder Client and Administrator users. The event that starts this initiator, StationAudioRequest, is generated by the Interaction Recorder Client and Administrator.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Station

Object ID

{all}

Notification Event

StationAudioRequest.

Outputs

Station Queue Identifier

The identifier for the workstation to which the .WAV file is played.

Audio File Name

The name of the .WAV file to be played. MP3 files are not supported.

Exit Paths

Start

This step always exits through the start path.

Process Automation Initiator

This initiator starts any time a handler is launched from a process. The output from this tool is used as input for the [Process Automation Send Handler Results](#) tool.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Outputs

Job ID

The job ID associated with the launching of the process.

Sequence ID

The sequence ID associated with the launching of the process.

PA Data Name

The name of the PA data element group.

Exit Paths

Start

This step always takes the Start exit path.

Provision Station Initiator

This initiator starts a handler that is part of the Auto Provisioning process. It is called when an unprovisioned station calls into the system and starts the auto provisioning process.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Station

Object ID

{all}

Notification Event

ProvisionStation

Outputs

Station Identifier

A string of the station ID. This value will be passed into the Bind Provisional Station tool.

Number being dialed

This field is not used in this release.

Call Requesting Pick-up

The Interaction ID of inbound station audio. This will be needed if you plan on playing audio to the call.

Exit Paths

Start

This step always exits through the start path.

Queue Period Report Statistics Initiator

The Queue Period Report Statistics Initiator generates an array of statistical information about a workgroup or user queue, or a Stats group. For more information about Stats groups, see [Report Tools](#). This initiator starts at the interval set in the **Queue/IVR reporting (min)** field in Interaction Administrator. For more information about this field, see [Configure reports for your CIC server](#) in Interaction Administrator help. This initiator requires that you set the StatServer_SendQPSNotification server parameter to a value of true. For more information about this server parameter, see [Optional General Server Parameters](#) in Interaction Administrator help.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Report Period Statistics

Object ID

Select {all}.

Notification Event

Select Queue Period Report.

Outputs

Queue Report Period Statistics Array

The identifier for the array passed in with the event.

Array Size

The number of elements passed in with the event.

Interval Started Date/Time

The date and time when the report period begins.

Interval Length in Seconds

The length of the interval covered by the report.

Array of Service Level Thresholds

These values represent the ranges used to determine the values for the liAllAnsweredBySvcLevels, liAllAbandonedBySvcLevels, liAcidAnsweredBySvcLevels, liAcidAbandonedBySvcLevels values that Queue Manager reports and are retrieved with the [Get Nth Period Statistics Report Data](#) tool.

Exit Paths

Next

This step always takes the Next exit path.

Receive Log Events Initiator

This initiator is triggered when an CIC log event is written to the MS Event Log. In order for this initiator to work properly, a registry value for each subsystem for which a user would like to get messages needs to be changed. The registry value is:

```
HKLM\Software\Interactive Intelligence\EIC\Remoco\Eic Subsystems\
```

There are several values at this key. Set the Info Log Event to 1 to allow the Informational messages for the subsystem to trigger the initiator. Set the Warning Log Event to 1 to allow Warnings to trigger the initiator.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

REMOCO

Object ID

Specify {all}

Notification Event

ReceiveLogEvent

Outputs

IC Server

The CIC Server that generated the Log Event.

IC Subsystem

The CIC Subsystem that generated the log event.

Event Type

- Informational
- Warning
- Error

Event Message ID

Maps to ID in Event Viewer.

Event Message

Maps to message in Event Viewer.

Substitution Strings

Strings that are inserted into the Event Message.

Exit Paths

Start

This step always exits through the start path.

Receive Trap Initiator

This initiator is triggered when an SNMP trap has been sent. These traps are defined in I3IC.MIB, which is located on the product iso in the folder \Additional Files\SNMP.

The two traps available are:

i3EicTrapRestart (trap 1) This trap is generated when an CIC Subsystem has been restarted.

i3EicTrapEventLog (trap 2) This trap is generated when CIC has written an entry to the Event Log on the CIC server accompanying MIB variable.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

REMOCO

Object ID

Specify {all}.

Notification Event

ReceiveTrap

Outputs

IC Server

The CIC Server that generated the trap.

Enterprise

Allows you to determine which Enterprise sent or generated the SNMP trap (could be Genesys, Dialogic, or Microsoft).

Enterprise OID

The number corresponding to the Enterprise. Genesys is 2793.

Generic Trap ID

6 means that this is an Enterprise specific trap.

Specific Trap ID

Either 1 or 2. See the trap 1 and trap 2 options described at the top of this topic.

Bound Variables

A list of any variables accompanying the trap. See the file I3IC.MIB in the location described above for more information.

Bound Values

The string values associated with the variable. See the file I3IC.MIB in the location described above for more information.

Exit Paths

Start

This step always exits through the start path.

Run Survey Initiator

This initiator starts when the agent disconnects a call and a customer satisfaction survey has been requested.

Note: This initiator will not be functional until a future service update.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

CCSurvey Object

Object ID

Select {all}.

Notification Event

Select {all}.

Outputs

Interaction ID

The unique ID assigned to the interaction.

Survey ID

The unique ID assigned to the survey for the interaction.

Secure Input Initiator

This initiator fires when Telephony Services starts a secure session.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Label

Secure Input

Outputs

Call Identifier

The identifier for the call on which the secure session is started.

Auxiliary Data Names

A list of labels.

Auxiliary Data Values

A list of names.

Transaction Id

The UUID that Telephony Services generates to identify the secure session.

Exit Paths

Start

This initiator always takes the Start exit path.

Send to Voice Mail Initiator

This initiator starts when a notifying call is sent to voice mail, such as when someone clicks the Voicemail button in a CIC Client or when someone manually transfers a call to another user's voicemail through the Transfer dialog box in a CIC Client. This is not the same initiator that starts the handler that records the voice mail button.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Call

Object ID

Select {all}.

Notification Event

Select Send To Voice Mail.

Outputs

Call Identifier

The unique identifier for a call being sent to voice mail.

Line Identifier

The line on which the call is currently connected.

Calling Party Number

The telephone number of the person who is being transferred to voice mail.

Calling Party Name

If available, the name of the person who is being transferred to voice mail.

Queue Identifier

QueueId is the unique identifier for the station from which this "send to voice mail" command has been placed.

Call transferred to voicemail?

True if the call was transferred, False if the call was sent by a user clicking the Voicemail button in a CIC Client. In the default handlers, a user's No Answer Message is not played when the call is transferred. This is so that the caller does not know if the call timed out on the queue or if the call was intentionally sent to voicemail from a CIC Client.

Exit Paths

Start

This step always exits through the start path.

SMS in Non System Queue Initiator

This initiator launches whenever an SMS Object enters a non-system queue.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

SMS Object

Object ID

Select {all}.

Notification Event

SMS in Non System Queue

Outputs

SMS Identifier

The unique identifier of the SMS Object.

Queue Identifier

The queue containing the SMS Object.

Exit Paths

Next

This step always takes the Next exit path.

SMS Monitor Initiator

This initiator launches whenever the specified SMS-related event occurs.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

SMS Object

Object ID

Select {all}.

Notification Event

Select either Created, Connected, Disconnected, Transferred, or Parked, depending on which event you want to monitor.

Outputs

SMS Identifier

The unique identifier of the SMS Object.

Exit Paths

Next

This step always takes the Next exit path.

SOAP Request Initiator

This initiator triggers if the 'Notification Event' of the request matches a specified string. The Notification Event on which the Initiator triggers is specified in the property dialog.

Outputs

SOAP Request

The handle representing the SOAP request. It can subsequently be used to query additional information from the (HTTP) header.

Initiator Event

String of the notification event that triggered the initiator.

SOAP Action

String. SOAP Action.

Exit Path

Start

This initiator always takes the Start exit path.

StatAlertServer Initiator

This initiator is called from Alert Server with the name of a handler to execute.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Outputs

Alert ID (string)

The ID of the alert that triggers the handler.

Alert Owner (string)

The owner of the alert.

Alert Owner Display Name (string)

The owner's display name.

Statistic ID

The ID of the statistic that triggers the alert.

Statistic Value (string)

The current value of the statistic.

Alert Description (string)

The user-provided description of the alert.

Alert Display String (string)

A display name for the alert.

Alert Reason (enum)

The enumeration for the reason that the alert is triggered.

Value	Description
0	Value equals
1	Value not equal
2	Value less than
3	Value greater than
4	Value enter/exit range
8	Value contains
9	Value does not contain
10	Value matched prefix
11	Value not available
12	Value available

Alert Data (string)

The user-supplied data provided for the alert.

Statistic Value Lower Bound (string)

The lower bound value for the statistic.

Statistic Value Upper Bound (string)

The upper bound value for the statistic.

Initiator for custom alert handlers (int)

A sequence ID supplied by StatAlert Server.

Station Off Hook initiator

This initiator starts anytime a station goes off hook. It provides a Station Identifier, Station Queue Identifier, and User Queue Identifier so that a handler can decide whether to play dial tone or assign an alerting or held call. This initiator is typically followed by the [Query Monitored Queues](#) and [Select Call](#) tools.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Station

Object ID

{all} makes this initiator work for all stations.

Notification Event

Select StationOffHook.

Outputs

Station Identifier

The station that went off hook.

Station Queue Identifier

The station queue associated with the Station Identifier.

User Queue Identifier

The user queue monitoring this station queue.

Exit Paths

Start

This step always exits through the start path.

Subroutine Initiator

The Subroutine Initiator is different from other initiators because it is started by a call from a handler or another subroutine, as opposed to an event that occurs on the CIC system. When you publish any subroutine, any parameters you create on the parameters page will become parameters in a subroutine tool that calls that subroutine. That subroutine tool will appear on the Subroutines page of the [Design palette](#). Refer to [Introduction to Subroutines](#) and [Add or edit a subroutine parameter](#) for more information on how to use this initiator.

Parameters

Label

The label for the parameter that appears in the properties notebook that this subroutine.

Variable

The variable that holds the value of what is passed into this parameter.

Type

The type of this variable, such as integer or string.

Input Only

Setting this parameter to true means that any change to the value of this variable will not be reflected in the handler that called it.

Setting this parameter to false means that a change to the value of this variable will be reflected in the handler that called it. In other words, the value can be changed and the changed value is passed back to the calling handler or subroutine.

Add button

This button opens the Subroutine Parameter dialog box for you to create a new variable. See [Add or edit a subroutine parameter](#) for more information on adding and editing subroutine parameters.

Edit button

This button opens the Subroutine Parameter dialog box for you to edit a selected variable. See [Add or edit a subroutine parameter](#) for more information on adding and editing subroutine parameters.

Delete button

This button deletes a selected subroutine parameter.

Exit Paths

Start

This step always exits through the start path.

Switchhook Flash Initiator

This initiator starts when a caller presses a flash button on a telephone or quickly depresses the switch hook. The Flash event can be generated from any station configured in CIC with a directly connected analog phone. For example, if a user wants to transfer the call from a stand-alone station, they quickly depress the station's switchhook. The call is put on hold and they are given a menu of choices of what to do. One of the options is to transfer the call.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Station

Object ID

Select {all}.

Notification Event

Select Flash.

Outputs

Station Identifier

The identifier for the station from which the switchhook flash originated.

Station Queue Identifier

The workstation queue associated with the station from which the switch hook flash originated.

Exit Paths

Start

This step always exits through the Start path.

Switchover Event Initiator

This initiator fires whenever a Switchover event occurs.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

SwitchoverService

Object ID

Select {all}.

Notification Event

Select either "Commence Operation" or "Discontinue Operation."

Outputs

Was Manual?

This Boolean indicates whether the Switchover event was Manual (True) or Automatic (False).

Exit Paths

Start

This step always exits through the start path.

System Initialization initiator

Use the System Initialization initiator in handlers that need to be executed when Interaction Processor (IP) subsystem starts.

If there are multiple handlers containing the System Initialization initiator when IP starts, IP will execute them all. The order of execution of those handlers is undefined. (If you have a set of procedures that requires a certain order, create subroutines and call them in order from a single System Initialization handler.)

This initiator has no input or output parameters.

Note: If you need to call a subroutine or your operations are going to be very long, sending a custom notification and doing all of the work in a Custom Notification Handler is best. IP waits for all System Initialization handlers to complete, so long tasks done with this can hold things up. The Custom Notifications are not processed until all handlers, including subroutines, have been loaded.

T1/E1 Wink Event Initiator

This initiator starts when a wink is received on a line, providing a way to start a handler for special processing when a wink is received. Lines are configured to receive winks in the spanti.prm file located in the \Dialogic\data directory. You can change the wink definition for a single span. If you want to change the definition of the wink for a single span, you must edit the file and rename it for the span. The name of the new file must then be entered for the particular span into the parameter field in the misc table of the DCM. Only one wink definition can be defined for each E1/T1 line.

Notes: This wink is not the wink used in wink start mode. Lines are configured to be either wink or immediate start in the line configuration under IA. What a wink looks like (wink definition) is in the spandti.prm file.

This initiator will not start when receiving the initial wink on an outbound call on a wink start line.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Call

Object ID

{all}

Outputs

Call Identifier

The identifier for the call on which the wink was sent.

Exit Paths

Start

This step always exits through the start path.

TCP/IP Connection Accepted initiator

When the [TCP Listen](#) tool detects a connection, it generates a TcpConnection event. This initiator is designed to detect that connection and start a handler. See [TCP/IP tools](#) for more information on using TCP/IP tools. See [Introduction to Initiators](#) for more information on using and configuring initiators.

Initiator Properties

Notification Object Type

TCP/IP Connection.

Object ID

Select {all}.

Notification Event

Select TcpConnection.

Outputs

Client Name

This tool queries the remote machine for a client name. If a name is not returned "unknown" is used.

Client IP Address

The IP address of the remote machine. This string value may look something like the following: "172.16.120.10."

TCP Port for connection

The port number for the connection.

Socket handle

The unique identifier for this established connection. Other TCP/IP tools use this handle for sending and receiving strings and integers.

Exit Paths

Start

This initiator always takes the Start exit path.

Timer Initiator

The timer initiator allows you to build a handler that runs at a time interval that you configure. You can configure the Timer initiator to start an instance of a handler every day, hour, minute or second.

Note: Timer initiated handlers will run on primary and backup servers.

Note: Timer initiated handlers will only run if published as a Primary handler and will not run if published as a Monitor handler.

Timer Page

IP Start Time

The Reference Time is the time around which the Run Every setting is based. The Reference Time is based on a twenty-four hour clock (midnight is 0 hours, 0 minutes, and 0 seconds).

For example, if you set the Reference Time to 1 hour and 13 minutes (01:13:00), and set the Run Every time to 30 minutes, then the handler will run every thirty minutes before and after the reference time. In this example, the handler would run at 00:13:00, 00:43:00, 01:13:00, 01:43:00, and so on). In all, the handler would run 48 times over the course of a day at thirty-minute intervals.

Check the **Start Time is UTC Time** checkbox if you want the time options to be calculated in Universal Coordinated Time.

Note: in order for a specific reference date to be selected (see below), the **Start Time is UTC Time** checkbox **must** be checked.

IP Reference Date

This can be used to have the timer only run on specific dates and/or days of the week. The default setting is "Handler Load," which means that the start time and time interval will be measured from the time the handler was first published. If this is not desired, you have the option of specifying a particular date, or choosing a day of the week as the reference date.

Setting a specific reference date tells IP to begin calculating the time as though the handler began running on that day or date.

Run Every

This is the interval at which the timer initiator runs. Set the days to 1, and it runs every day. Set the days to 2, and it runs every other day. See the example above for information on how this setting interacts with the Reference Time setting.

If you want a handler to run on the first of every month or year, or on a specific day of the month, you can use a DateTimeNow operation to assign the current date and time as the value of a variable. Then use a Condition step to determine whether the current day of month is equal to one. If it is not, the handler ends. Configure the Timer initiator for this handler to run every day, and you'll have a handler that does something on a specific day of the month.

Show UTC Times

Check this box if you want the times to be shown in Universal Coordinated Time notation.

Exit Paths

Start

This step always exits through the start path.

Transfer Request Initiator

This initiator begins when a user attempts to transfer a call to another queue or an external number.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Call

Object ID

Select {all}.

Notification Event

Select TransferCallRequest.

Outputs

Call Identifier

The unique identifier for a call that is being transferred.

Number being dialed

The telephone number of the transfer recipient. A comma causes a two-second pause, and any numbers after the "/" symbol are dialed after the call is connected.

Base number being dialed

The telephone number with any numbers after the "/" symbol removed. For example, if a user calls 1-317-872-3000/103, the base number would be 1-317-872-3000. The base number can be used for security lookups.

Line Groups

The line groups on which this transfer will be placed.

Station Queue Identifier

The identifier for the station queue from which this transfer was placed.

User Queue Identifier

The name of the user queue from which this transfer was placed.

Use Putback (if available)

This Boolean, set appropriately by default by CIC according to the configured line's support for putting a call back on the originating system, is used to determine whether or not to use the putback feature if it is available for that line. If putback is available, then setting this parameter to True will cause the transfer to be attempted using the Putback operation. If the Putback operation fails within TS Server, a conventional transfer will be attempted.

Note: We strongly recommend that you do not change the default value of this parameter unless you know exactly when and why you need to override the default system value. This value is normally passed through the Transfer Request initiator to the Complete External Blind Transfer too.

Exit Paths

Start

This step always exits through the start path.

Transfer to System Queue Initiator

The purpose of the Transfer to System Queue initiator is to start the CallOfferingSystemQueue handler and pass along information to that handler.

Initiator Properties

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Call

Object ID

All

Notification Event

Select Transfer to System Queue.

Outputs

Call Identifier

The identifier for the object that started this handler. A Call Identifier is a value that stays the same throughout the life of a call.

Line Identifier

The identifier for the line that this call object came in on. The contents of this variable correspond to one of the lines set up in Interaction Administrator. Line identifier could be null if the call is an intercom call.

Called Party Number

The telephone number the caller dialed.

Calling Party Number

If available, the number of the calling party.

Calling Party Name

If available, the name of the calling party.

Exit Paths

Start

This step always exits through the start path.

Transfer Conversation Initiator

This initiator is fired when a social media conversation interaction is transferred to a non-system queue. The `System_InteractionOnNonSystemQueue` handler is launched with this initiator.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Interaction Object

Object ID

Select {all}.

Notification Event

Interaction on Non System Queue.

Outputs

Interaction ID

The unique identifier for the interaction.

Queue Identifier

The queue to which the interaction was transferred.

Exit Paths

Next

This step always takes the Next exit path.

Transfer Direct Message Initiator

This initiator is fired when a social media direct message interaction is transferred to a non-system queue. The `System_InteractionOnNonSystemQueue` handler is launched with this initiator.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Interaction Object

Object ID

Select {all}.

Notification Event

Interaction on Non System Queue.

Outputs

Interaction ID

The unique identifier for the interaction.

Queue Identifier

The queue to which the interaction was transferred.

Exit Paths

Next

This step always takes the Next exit path.

UMF Message Received Initiator

This initiator allows the author to type a string value for both the event ID and Object ID. The initiator will then trigger when a notification with matching event and object IDs is received. The same initiator receives messages sent as events and requests. In the latter case, certain outputs that contain return information for a response will contain valid values. If the message was sent as an event, those outputs will be zero.

Inputs

Notification Event

A string value that matches the Event ID you set in [UMF Send Event](#) or [UMF Send Request](#).

Object ID

An integer value that matches the Object ID you set in UMF Send Event or UMF Send Request.

Outputs

Message Handle

A valid handle to the incoming message from which the data elements are to be read.

Response Correlation ID

A unique identifier for this message if the message was sent synchronously. Otherwise, this value will be zero. If the message requires a response, the value of the variable will contain a non-zero value. This value must be passed unchanged to the [UMF Send Response](#) tool.

User Status Monitor Initiator

This initiator watches for an event generated by a CIC Client user changing his or her status. For example, a user might change their status from Available to Do Not Disturb.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

User

Object ID

Select {all}.

Notification Event

Select Status.

Outputs

User ID

The user id of the agent who changed his or her status.

First Name

The first name of the agent who changed his or her status.

Last Name

The last name of the agent who changed his or her status.

Status Text

The current status that resulted from the change. This might be 'Available' or 'Do Not Disturb'.

Status Key Text

An unlocalized version of the status message. In a future release of CIC, there will be a localization mapping table for each of the keys. This will be used to display the appropriate string to an end user based on the user's Local settings.

Status Change DateTime

A date time value indicating when the status changed.

Date String

The date information contained in the status, such as a return date.

Time String

The time information contained in the status, such as a return time.

Extension

The agent's extension.

DND Indicator

The Do Not Disturb indicator. True if the status is a DND status. False if the status is not a DND status, like Available.

Logged In Indicator

A value of 1 (one) will be returned if the user is logged in, and a value of 0 (zero) if the user is not logged in.

From Server Flag

This output parameter is for internal user only, and it may be removed in a future release.

Station Id

The station Id where the agent changed his or her status.

Error Flag

Any error information passed in with the event.

DND Indicator

Indicates whether this status is configured as a DND status in Interaction Administrator.

ACD Indicator

Indicates whether this status is configured as allowing ACD calls in Interaction Administrator.

Forward Indicator

Indicates whether this status is configured as a Forward status in Interaction Administrator.

On Phone Indicator

Indicates whether this status is configured as an On Phone status in Interaction Administrator.

Previous Status Key

An unlocalized version of the previous status message. In a future release of CIC, there will be a localization mapping table for each of the keys. This will be used to display the appropriate string to an end user based on the user's Local settings.

Previous Status

The user's previous status.

Previous Status Change Datetime

The time when the status was changed before the most recent change.

Exit Paths

Start

This step always exits through the start path.

Work Item Transferred to Queue Initiator

Initiator Properties

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Workflow Work Item Object

Object ID

All

Notification Event

Work Item on Non System Queue

Outputs

Interaction ID

The identifier for the object that started this handler. A Call Identifier is a value that stays the same throughout the life of a call.

Queue Identifier

The identifier for the line that this call object came in on. The contents of this variable correspond to one of the lines set up in Interaction Administrator. Line identifier could be null if the call is an intercom call.

Called Party Number

The telephone number the caller dialed.

Calling Party Number

If available, the number of the calling party.

Calling Party Name

If available, the name of the calling party.

Exit Paths

Start

This step always exits through the start path.

Wrapup Required Initiator

This initiator is for internal use only.

System_IncomingSocialMediaConversation

This initiator is for all the Social Media Conversations. This initiator is fired when a social media conversation interaction is transferred to a non-system queue. The System_IncomingSocialMediaConversation handler is launched with this initiator.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Interaction Object

Object ID

Select {all}.

Notification Event

Interaction on Non System Queue.

Outputs

Interaction ID

The unique identifier for the interaction.

Queue Identifier

The queue to which the interaction was transferred.

Exit Paths

Next

This step always takes the Next exit path.

System_IncomingSocialMediaDirectMessage

This initiator is for all the Social Media Direct Messages. This initiator is fired when a social media direct message interaction is transferred to a non-system queue. The System_IncomingSocialMediaDirectMessage handler is launched with this initiator.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Interaction Object

Object ID

Select {all}.

Notification Event

Interaction on Non System Queue.

Outputs

Interaction ID

The unique identifier for the interaction.

Queue Identifier

The queue to which the interaction was transferred.

Exit Paths

Next

This step always takes the Next exit path.

Procedures

Starting Interaction Designer with command line parameters

The following command line parameters can be used when starting Interaction Designer.

Command Line Parameter	Description
/HandlerOpenAutoXMLPath={location where .xml files will be created}	Creates an xml version of each handler opened in the specified directory. For example, if Designer is launched with: /HandlerOpenAutoXMLPath=c:\temp and then the handler "MyHandler1.ihd" is opened, Designer automatically generates a file named "MyHandler1.xml" in c:\temp which contains the XML for MyHandler1 when the handler was opened.
/intermediatepublish:{name of text file that contains list of .ihd files for intermediate publish}	Generates an .i3pub file for each .ihd file in the text file.
/notifier={server name}	Tells Interaction Designer which server to use as the CIC server for publishing, debugging, and other operations which requiring a connection to an CIC server.
/publish:{name of text file that contains list of .ihd files to be published}	Tells Interaction Designer to open each .IHD file in the text file and attempt to publish it. If all the handlers are published successfully, then Interaction Designer will automatically close itself. Otherwise, it will remain open with the failing handlers open.
/ResetWindowPos	Resets windows to original default size and position.
/RunMinimized	Opens Interaction Designer with the main window minimized.
/ToolReport:{filename}	Generates a text file containing a listing of tools by category (tab), showing the tool (or initiator) label followed by modulename::toolname. It can also help you figure out which dll & src dir a particular tool came from.

Working with Handlers and Subroutines

Overview of Building and Modifying Handlers and Subroutines

This section offers an overview of the entire process of building and editing handlers and subroutines. You should read this before you begin using Interaction Designer to build or modify handlers and subroutines. This section contains the following subsections:

- Overview of Steps
- Step 1: Research
- Step 2: Design
- Step 3: Building
- Step 4: Activating
- Step 5: Testing
- Step 6: Promoting

Overview of Steps

The process of building handlers can be broken down into six steps.

Step one: Research. You should be familiar with the handlers and subroutines currently running on your system.

Step two: Design. Map out the functionality you plan to build, and figure out how it will run with the other handlers and subroutines.

Step three: Building. Begin creating and linking steps to perform the new functionality.

Step four: Activating. When you finish building the handler or subroutine, you must compile and move it to the IC server using the automated Publish process. Handlers and subroutines are published through Interaction Designer.

Step five: Testing. Test the handler on your live system or on a second system you've set up for testing purposes.

Step six: Promoting. If you are not already using the handler on your live system, activate the handler.

Step One: Research

The first step in building handlers and subroutines is research. You must be familiar with the handlers and subroutines currently being used on your CIC server. You need to know if the handler or subroutine you're planning can be integrated into a currently running handler. Also, find out if you need to build a new handler or subroutine, or modify an existing one. You cannot build a new handler or subroutine if you don't spend some time looking at the existing handlers.

There are several places you can look for information on the existing handlers. First, look at the contents of the Handler Help menu. Many of the handlers and subroutines shipped with CIC are documented under Handler Help. This documentation summarizes the functionality of the handlers and subroutines, and can help you see where you might want to make customizations.

Also, review the reference and procedural topics in this online help system. The tool documentation in the Reference section can help you learn the function of each step in a handler or subroutine. The procedural topics and various introductions can help you understand how handlers work.

Step Two: Planning

Once you are familiar with how the handlers and subroutines are working on the CIC server, begin planning how to build in the new functionality. Decide how the new functionality will be implemented. Review the following scenarios when making your decision.

When should I modify an existing handler or subroutine?

In many cases, modifying an existing handler is the best way to implement new functionality. In a simple example, you might want to add a new key-press option for callers using your IVR (Interactive Voice Response) system. In this case, adding new functionality only involves recording a new prompt, adding a condition to a Selection step, and adding a few other steps to your existing IVR handler. You don't need to write a whole new subroutine or handler to add this simple functionality. If you only need small changes or additions to the CIC functionality, consider small modifications to your handlers or subroutines.

If you decide to modify an existing handler or subroutine, we recommend that you only do so using the [customization points](#) provided. Customization points ease future upgrades. Modifications made outside these customization points may be overwritten when a hotfix or service release is installed. Modifying handlers outside the designated customization points may also cause undesired behavior in CIC.

When should I build a new handler or subroutine?

If you are planning a significant addition or change in functionality, you probably need to build a new handler or subroutines. In most cases, you should build a subroutine. A handler will start when an event occurs on the server to start that handler. Most of the events that occur on the CIC server already have a handler associated with them. Unless you are adding third-party products that generate their own unique events, you build a subroutine.

An example of a piece of third-party equipment that might generate a unique event is some kind of machine monitor. Suppose a hospital has a computer monitoring a piece of equipment. Anytime that equipment fails, the monitoring computer creates an alarm event. This event could be passed into CIC and start a handler designed to watch for an alarm event.

Whether you decide to build a handler or a subroutine, map out the new functionality on a whiteboard. Try to think in terms of the Interaction Designer tools, and what groups of tools create the needed logic. Remember that creating handlers is programming; planning at the outset saves time when you start building the handler.

Step Three: Building

Once you've mapped out the functionality and decided to implement the functionality, you're ready to start modifying or building your handler or subroutines in Interaction Designer. Use Interaction Designer to add new steps and modify existing steps. For help on a particular step, click the help button on its properties page, or you can select a tool on the tool palette and press the F1 key. The help for each tool consists of a general description of the step, its parameters, and its exits. You can also find help on working with steps from the contents page of the online help. Finally, look at how steps are used in other handlers and subroutines.

You do not need to deactivate a handler before you begin editing because any changes you make to a handler or subroutine do not take effect until you publish the handler or subroutine and activate that handler or subroutine using the Manage Handlers notebook.

Step Four: Activating

Once you have finished building or modifying a handler or subroutine, you need to publish it so that your changes take effect. Publishing a handler is the automated process of converting the handler to Java code, compiling that code with a Java compiler, and placing the compiled code on the CIC server. During the publishing process, you may be notified of errors in the handler or subroutine. These errors might result from unfilled parameters, invalid variable names, or other problems. Refer to the publishing online help for advice on successful publishing. You may also need to consult the documentation that accompanied the Java compiler.

Subroutines must be published before they can be called from other handlers and subroutines. Once you have published a new subroutine, a new subroutine tool appears on your subroutine palette. Use this tool to create a new step in the handler or subroutine that calls the new subroutine.

There are certain situations where you may not want to publish your handlers or subroutines to the same server where you created them. To accomplish this, an [intermediate publish](#) file is generated and saved to a specified location, where it can then be moved to the desired server and the publishing process completed.

When a handler or subroutine is first published, you have the option to have it made active immediately. If you choose not to do this, or if the handler has been imported from another server or previously made inactive, you will need to activate it in the Manage Handlers notebook before it can be used in CIC.

Step Five: Testing

After you've published your handler or subroutine, either to a test server or the production server, test the functionality you've built.

Step Six: Promoting

If you've not already set a handler to active using the [Manage Handler notebook](#), you should do so. This activates the new functionality.

Related Topics

[Handler Best Practices](#)

[Publish a handler or subroutine](#)

Introduction to Handlers

Handlers are the programs that define the interaction processing functionality of the CIC system. A handler is a group of actions (steps) that start when an event occurs on the CIC system. For example, the IVR (Interactive Voice Response) system that callers interact with is a handler. Handlers record and send voice mail. Almost every call processing action that occurs on the CIC server is the result of a handler. Handlers also define the functionality of Internet chat requests, HTML document generation, ACD (Automated Communication Distribution), and more.

What is an event?

Events are the things that happen to [objects](#). Every time one of the modules in CIC does something to an object, it generates an event. For example, when [Telephony Services](#) receives an incoming call, it creates a new call object with a unique Object ID. Telephony Services then generates an event called Incoming Call. Events can even be generated by hardware or software that are not part of the CIC system (for example, a security card reader or a computer that monitors a piece of equipment).

Events serve two purposes in CIC. First, an event can tell [Interaction Processor](#) to start an instance of a handler designed to respond to that event. When Telephony Services generates an Incoming Call event, the [Notifier](#) tells Interaction Processor that the event has occurred. Interaction Processor starts an instance of any handlers configured to run when the Incoming Call event occurs. Second, an event can carry information about the event. For example, an event generated by an incoming call carries information about that call, such as the caller ID and the name of the line on which that the call is coming in. The information contained in the event can be used by the handler to make decisions about what to do with the object.

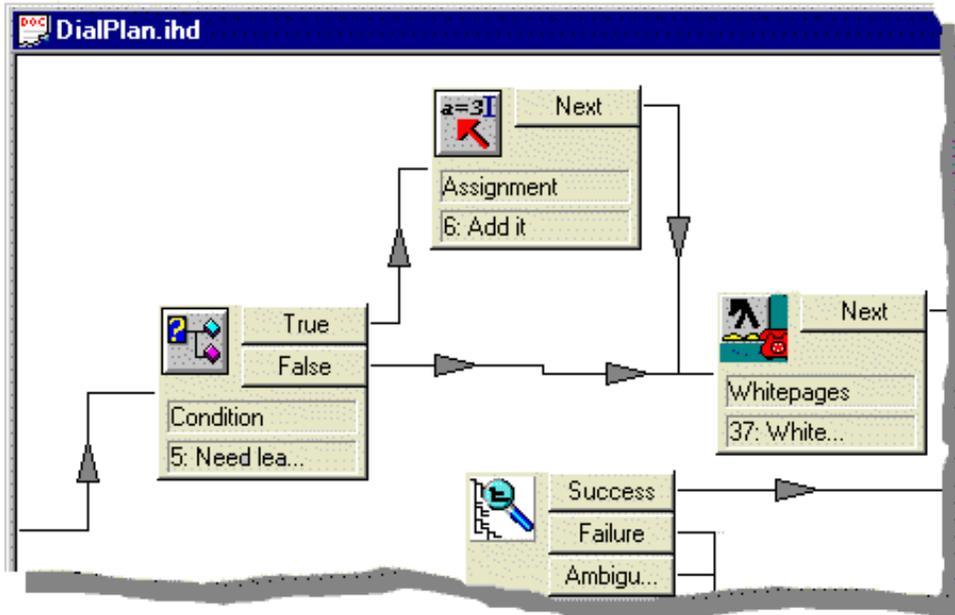
How does an event start a handler?

Each event is of a unique type, such as incoming call or chat session request. Each handler has an initiator that watches for an event of a unique type to occur on an object of a unique type. An initiator can even be configured to start only for objects with a specific Object ID. (If you view the properties of an initiator, you'll see that one of the parameters is the type of event that will start

the handler, one is the object type, and one is the Object ID.) When an event is generated, Interaction Processor starts any handler whose initiator is configured to start when that event occurs.

How do handlers work?

Like a program, handlers perform a series of actions. Handlers look like flow charts (see the following figure), and each action in the handler is graphically represented by a [step](#). These steps are performed in an order determined by how the steps are linked together. Different steps perform different actions, and the combination of actions determine what happens when the handler runs. Some steps start other handlers, called [subroutines](#).



Interaction Designer provides all the basic programming controls for you to build your handlers. Using steps, you can assign variables, build conditions, test conditions, create selection statements, assign variable types, and perform other useful actions.

For a description of the purpose and properties of each step, select that step and click F1.

Choose one of the topics below for instructions on how to work with handlers:

[Create a new handler](#)

[Introduction to steps](#)

[Introduction to subroutines](#)

[Open an existing handler](#)

[Print a handler](#)

[Publish a handler](#)

Create a new handler or subroutine

Create a new handler when you want to perform actions in response to an event that occurs on the CIC server. Before you create a new handler, review the documentation for the handlers that are already running on your system. The event that you want to start your new handler may already be starting a previously published handler. In this case, you may want to build a subroutine.

- From the File menu, choose New.

or



- Click the  button on the toolbar. A new Handler window appears.

Note: you may also add your own documentation for any handlers you create to Designer's Help menu. To add an item to Designer's Help menu, create a Help folder on the server under `..\I3\IC\Server`, and place your help folder or files there. You can create help files of any type, as long as there is an associated application. The next time Interaction Designer is started, anything located in that directory will appear at the bottom of the Help menu.

Related Topics

[Introduction to Handlers](#)

[Introduction to Initiators](#)

[Introduction to Subroutines](#)

[Overview of Building Handlers and Subroutines](#)

Handler Standards

This topic describes the standards that were used in creating the handlers that shipped with CIC. We highly recommend adopting these standards when creating your own handlers.

Variables (locally scoped)

Variables have been named using a convention similar to Hungarian notation. Each variable name has a prefix indicating the data type in lowercase followed by the variable name. Except for native data types, many relate to data structures and so are unique to handlers.

The following table outlines data type prefixes for native data types.

<u>Data Type</u>	<u>Notation</u>	<u>Example</u>
Integer	i	iCount
Numeric (double)	n	nPayRate
Boolean	b	bSuccess
String	S	sName

The following table lists the data type prefixes for external data types. For each of these types there is a corresponding list data type. Any list data type starts with an l (lowercase L), followed by the type abbreviation. For example, the Mail Folder data type (e.g., mfSysAdminAccounts) has a corresponding List Mail Folder data type (e.g., lmfSysAdminAccounts).

<u>Data Type</u>	<u>Notation</u>	<u>Example</u>
Accumulator Lock Key	aclockkey	aclockkeyTimeout
aQueuePeriodStatistics	aqprdstat	aqprdstatTimeout
Calendar ID	cal	calApril

Call ID	Interaction	Interaction1
Conference ID	conf	confFollowMeCall
Database	db	dbCustomerData
Date Time	dt	dtTomorrow
DB Connection	dbconn	dbconnSQLSever
Diagnostic Data	dd	ddErrorMsg
EntryHdl	enth	enthTemp
EntryListHdl	entlsth	entlsthTemp
FaxEnvelopeID	faxenv	faxenvSend
FaxFileId	faxfile	faxfileIncomingFax
FaxObjectId	faxobj	faxobjTempFax
FaxPageListId	faxpglst	faxpglstProductListing
File Handle	fileh	filehTempData
Host Interface Connection	hst	hstMainFrame
Lock Handle	lckh	lckhThisLock
LoginHdl	logh	loghICUser
Mail Folder	mf	mfSentItems
MQ Connection	mqcon	mqconReportingQueue
MQ Object	mqobj	mqobjReortingObject
OperationListHdl	oplsth	oplsthTemp
Prompt	prmp	prmpHello
Recording	rec	recConfCall
RequestHandle	reqh	reqhTemp
Semaphore Lock Handle	semlckh	semlckhTemp
Sequence	seq	seqTemp
Session	sess	sessTemp
SessionHdl	ses	sesTemp
SOAP Request	soap	soapTemp
SocketHandle	sockh	sockhTCPConenction
String	s	sName
UMF Message	umf	umfIncoming
Web Connection	webconn	webconnCustomerOrder
XML Node	xmlnode	xmlnodeUserInfo
XML Node Iterator	xmlnodeit	xmlnodeitProducts

Note that the Call ID variable does not follow the normal convention. This is intentional. Because all the xIC product lines focus on processing interactions, this one variable is spelled out.

Variables (globally scoped)

Any globally scoped variables starts with `g_` to denote that it is global, and are followed by the lowercase data type indicator. For example: `g_iUsersLoggedIn`

GET and SET subroutines should be written for each global variable. This is because handlers that access a global variable place a lock on that variable for the time period that the handler is in scope. By creating these separate subroutines, the lock will only be on the global variable for the shortest possible amount of time, allowing other handlers to access it more easily.

Subroutines

All handlers that start with an initiator other than the subroutine initiator should be named with this prefix: "System_" In addition, the name of the handler should be in a mixed case, capitalizing the first character of each word. For example: `System_IncomingCall.ihd`

Any parameter (either input or output) to a subroutine should be named with a "p_" prefix. This allows anyone looking at the handler to immediately know that this variable is a parameter to a subroutine. For example: `p_Interaction1`

Each subroutine should have a Boolean variable defined as `bTransferred` that is set to show whether the interaction is terminated in this handler (False) or continues on (True).

Prompts

All fields within the Prompt Editor should be filled out: Prompt Name, Description, Chat Text, TTY Text.

In naming prompts, the names should be written in all uppercase with a suffix containing the language abbreviation. For example, `HELLOWORLD_ENUS`.

General Layout

Handlers with tightly grouped or overlapping steps, links going in seemingly random directions, and unused tools are very hard for others to read and debug. Sloppy handlers are also more likely to contain errors. While we do not want to dictate specific logic flow, we do highly recommend following these general guidelines concerning layout:

- Never allow links to flow through a tool step. Either move the steps to make more space, or re-route the link lines so that they go around, instead of through, the intervening steps.
- Do not bunch tools together; make sure there is plenty of white space between the tools. If you are modifying a handler and need to make room for new steps, drag out the other tools to make space; do not just squeeze the tools in. Make use of the zoom feature to grab and move large groups of tools if you need to make room for new steps.
- All label fields on tool steps should be filled out. For example, assignment steps should have the variable name they are assigning in the label, like "sUserName." Condition steps should give some indication of what is being evaluated, such as "Success=TRUE?" or "iCount >= # of Users?," etc.
- Drag all tool steps out wide enough that the Tool Name and Tool label are completely visible. The only exceptions to this might be the selection tool that could contain very wide case elements, but even this can be widened completely in most cases.
- Align your steps. If you have a number of tool steps that flow right to left, up to down, or whatever direction, align them so they look nice. The alignment buttons in the layout toolbar make this a simple task.
- Add notes as needed. If you are doing something that might seem odd or ambiguous to others, fill out the notes field to explain what you are doing. For example, a note might explain that a set of tools is intended to work around a bug in LDAP. Prefix the tool's label with the word, "NOTE:" to let others know that there is more information in the tool's properties page.
- The description field of initiators should list any customization points in that handler.
- When you are finished developing a handler, ensure only the variables being used are present in the handler by selecting all of the unused variables in the Variable viewer and deleting them.

Save a handler or subroutine

Whenever a handler contains unsaved changes, Interaction Designer will indicate this by placing an asterisk (*) after the handler's name in the title bar. To save the changes to a handler or subroutine:

1. From the File menu, click Save.
or



Click the button on the toolbar.

2. If the Save As dialog box appears, give your handler a name and click OK. We recommend that handler names contain only alphanumeric characters and underscores. I.e., names containing special characters and spaces are not recommended.

Handler and subroutine files use an .IHD extension.

Note: You cannot save two or more handlers with the same name in the same location. If you want to save a handler with the same name as another handler currently open in Interaction Designer, you must save that handler to a different location.

Publish a handler or subroutine

You must publish a handler or subroutine to make it available to run on the CIC server.

The initiator in a handler is configured for a specific event to occur to a specific object. During the publish process, Interaction Processor tells the Notifier that it needs to tell Interaction Processor each time the specific event occurs to the specific object. When Notifier tells Interaction Processor that the event has occurred, Interaction Processor starts an instance of that handler. See [Introduction to Initiators](#) for more information.

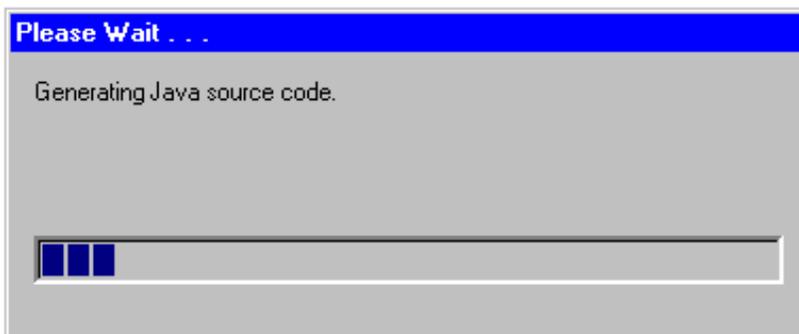
Note: Be sure to back up copies of your current handlers and subroutines before publishing new versions. This is especially important if you are overwriting existing handlers and subroutines with modified handlers of the same name. Also, you must create and publish your handlers and subroutines on a computer connected to the CIC server on which the handlers or subroutines run.

To publish a handler or subroutine:

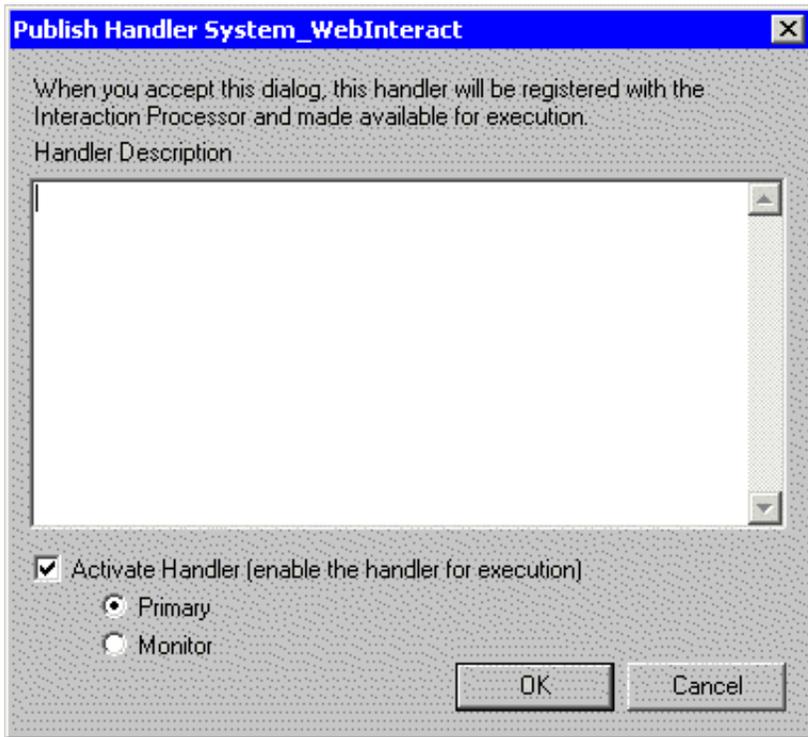
Use this procedure after you have created or modified a handler and are ready to begin using the handler on the CIC server. Be sure to back up the source code for any currently running handlers of the same name.

1. From the File menu, choose Publish.

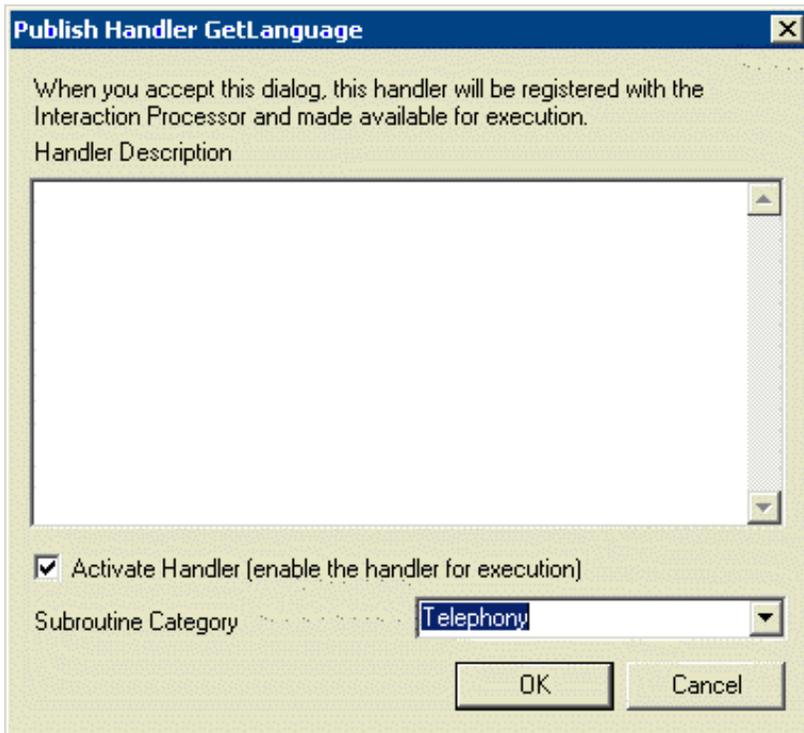
The Please Wait dialog box appears. This dialog box displays the current status of the Publish operation, as shown in the following figure.



2. Type a description of your handler in the Publish Handler dialog box that appears. If you want the handler to be active upon being published, select the Activate Handler check box and select whether the handler is a Primary or Monitor handler.

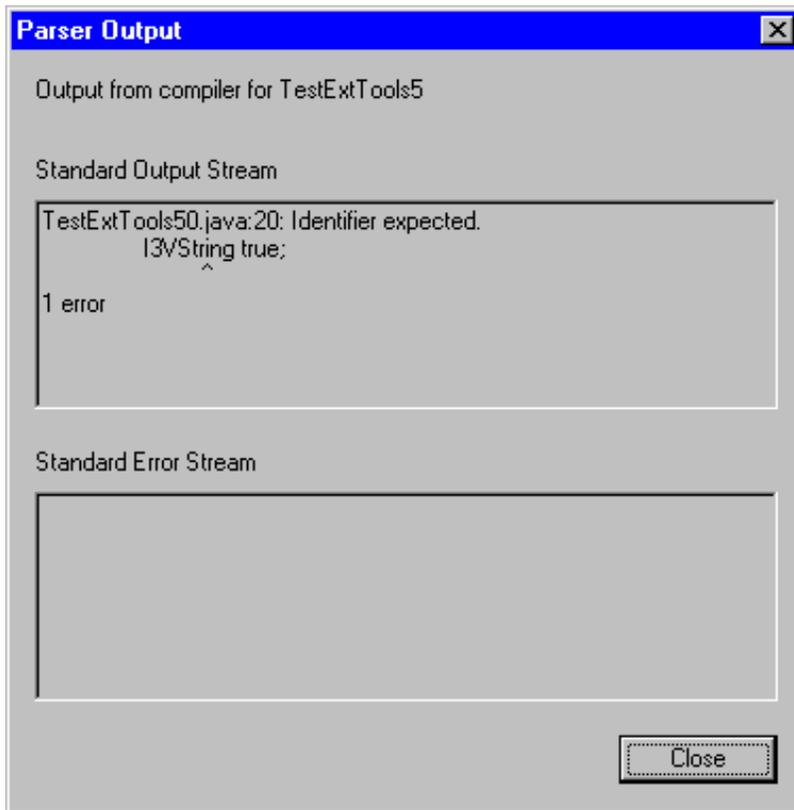


3. If you are publishing a subroutine, type a new subroutine category or select an existing subroutine category. These categories represent tabs on the [design palette](#), and typing a new category name creates a new page in the subroutine category field.



4. Click the OK button in the Publish Handler dialog box to continue the publish process.

If the publish operation is successful, the Please Wait dialog box disappears. If there are errors, the Parser Output dialog box appears, as shown in the figure below.



Important notes about publishing:

- **Backup Copies:**
A copy of the handler or subroutine (.IHD file) you are publishing is saved in CIC Directory Services. You can recover this file by doing a search for the file on the server (it will have an extension of ".ihd."), and copying this file to another location. Then open the file using Interaction Designer. Only the last published copy is saved, and previous copies are overwritten during the publishing process. If you want to save older copies of handlers, you must save them manually to a backup directory.
- **Publishing Rights:**
You must have handler publishing rights given in the Interaction Administrator, or you will receive an error when you try to publish your handler. Contact the CIC system administrator to obtain publishing rights.

Also see the [Batch Publishing](#) topic for advanced options in publishing handlers.

Publishing a handler created in a previous release of CIC

When you open a CIC handler that you wrote or modified in a previous release of CIC, you may receive Interaction Designer error messages or warnings indicating that a tool's parameters and/or exit paths have changed since the previous release. An error message will tell you the offending tool's Step Label and Node ID, and attempt to explain the problem with an error message.

You cannot publish this handler until you correct the problem and save it in the newer release of CIC. Use the Step Label, Node ID, and error message text to locate the step and diagnose the problem. Make any changes that are necessary to solve the problem, such as creating a variable to contain the value of the parameter, or creating a link from the new exit path. Then save the handler and try publishing it again.

If you are still unable to solve the problem, contact technical support.

Intermediate Publish

Generating an .i3pub file is a means of breaking the publishing processes into two stages so that a handler can be assembled, packaged into an intermediate format and saved to disk, then transferred and published on an entirely different server. This intermediate file contains the .class file, the configuration data for DS, the audio prompt data, and a copy of the .ihd file, along with everything else needed for publish.

When this process is started, the user will be prompted for a directory into which the intermediate file will be saved. After making this selection, the publish dialog box will appear which will prompt the user to enter a handler description and handler activation data. Interaction Designer then converts the handler, compiles the code, and places everything needed for complete publish in a file with the extension .i3pub in the specified directory.

When you select **Generate .i3pub File** from the **File** menu, the following dialog box appears:



Enter a description of the handler in the text window provided. If you want the handler to become active as soon as the publish is complete, select the **Activate Handler** check box at the bottom of the dialog box. If **Activate Handler** is selected, you must also select whether the handler is going to be a Primary or Monitor type handler. Follow these links to review the differences between [Primary](#) and [Monitor](#) handlers.

When generating an .i3pub file for a subroutine, the dialog box looks slightly different than for a handler:



Add the appropriate notes in the text window and select whether or not you want the subroutine to become active immediately upon publish, just as you would with an event-initiated handler. Also, you must select a category for the subroutine.

It is important to note that selecting the activation status of a handler will change any activation status the handler may have previously had. For example, if a handler was Active Primary before, and someone republishes it and selects Active Monitor, then that handler will be changed from Primary to Monitor. Interaction Designer attempts to help the user here by setting the default to the condition that existed before – if a previously published handler was active, then the checkbox will appear already checked; if it was Primary, the Primary option will be set, etc. If the handler had not been published before, then it defaults to Inactive.

This "activate on publish" feature also applies to [batch publish](#). Batch publish allows a user to create a text file listing .ihd files to be published. The user can then start Interaction Designer with a command line parameter (/publish:{filename}) and Interaction Designer will open the text file, then open each .ihd file listed and publish it. The user can also optionally specify the activation status in the text file as well. Consider the following example batch publish file:

```
FirstOne.ihd
SecondOne.ihd | Primary
ThirdOne.ihd | Monitor
FourthOne.ihd
```

In this example, after FirstOne is published, there will be no change to its activation status (if it was Primary before, it's still Primary, etc.) SecondOne will be set Active Primary, ThirdOne will be set Active Monitor, and FourthOne will have no change to its existing activation status. If FirstOne or FourthOne had no activation status before (i.e. first publish on this server), then the default status will be Inactive.

Using .i3pub Files

A separate executable is shipped with CIC called [EICPublisherU.exe](#) that allows users to install the i3pub files without the need for Interaction Designer. EICPublisherU is a simple command-line utility that can read .i3pub files and finish publishing them to the current CIC server. In addition to publishing, EICPublisherU also activates the handler as selected by the Interaction Designer user who created the intermediate file. Users can simply double-click on an .i3pub file and EICPublisherU runs automatically to publish the .i3pub file to their server.

For example, a reseller can make changes to a handler, generate an .i3pub file, and then make an install program that includes the i3pub file. When their customer runs the install program, it executes EICPublisherU to publish the .i3pub file.

EICPublisher

EICPublisher is a simple command-line utility that can read .i3pub files and "finish publishing" them to the current CIC server. EICPublisher is a separate executable that is shipped with Interaction Designer.

Note: EICPublisher 2.3 will not publish .i3pub files generated with any previous release of Interaction Designer. Only .i3pub files generated with Interaction Designer 2.3 can be published with EICPublisher 2.3.

Usage:

To use EICPublisherU, enter the following command:

```
eicpublisheru [/help] [/?] [/verbose] [/noprompts] [/notifier={hostname}] [/user={username}]  
[/password={password}] {filespec | @{file}}
```

where

/help or **/?** causes display of help text

/verbose causes the program to output extra process status info to stdout

/noprompts causes the program to avoid prompting the user for any input

{hostname} specifies a target CIC server

{username} specifies a valid CIC user id that has rights to publish on the target CIC server

{password} specifies the password for {username}

{filespec} specifies the file(s) to be processed

@{file} specifies the a text file listing the files to be processed

The [] means that piece is optional. Note that the username must have "Publish and Manage Handlers" rights (granted through Interaction Administrator).

Examples:

<pre>Eicpublisheru /notifier=WhiteSands /user=JohnSmith /password=smithj FirstOne.i3pub</pre>	<p>Publishes FirstOne.i3pub to WhiteSands.</p>
<pre>Eicpublisheru /notifier=WhiteSands /user=JohnSmith /password=smithj c:\temp\First*.i3pub</pre>	<p>Publishes all files that fit the mask "First*.i3pub" from the c:\temp directory.</p>
<pre>Eicpublisheru /notifier=WhiteSands /user=JohnSmith /password=smithj @somefile.txt</pre>	<p>Opens the file somefile.txt and publishes the intermediate files listed in there (one per line). So somefile.txt could contain:</p> <pre>c:\temp\FirstOne.i3pub SecondOne.someotherextension ThirdOne.i3pub</pre>

Note: The files don't have to have .i3pub extensions – EICPublisher does not require files to have any specific extensions. When Interaction Designer generates the .i3pub file, it uses the same base name as the handler document. For example, if you have FirstOne.ihd open, and you generate an .i3pub file from it, you'll get FirstOne.i3pub. You can rename that file to anything you want, and EICPublisher will still process it. But the real handler name (FirstOne) is still contained in the .i3pub file, and that's the name that will be used when it's published, not the name of the .i3pub file.

Example:

Suppose you start Interaction Designer, open FirstOne.ihd, generate FirstOne.i3pub, then rename it to NotTheFirstOne.junk. You send NotTheFirstOne.junk to a customer and they run EICPublisherU on NotTheFirstOne.junk to publish it to their server. After processing is complete, the handler FirstOne will exist on the customer's CIC server, *not* NotTheFirstOne.

Batch Publishing

Batch publishing is a way of publishing a group of handlers all at once, rather than one at a time through Interaction Designer. By default, batch publishing publishes all of the default packaged handlers located in the `i3\ic\handlers\40handlers` directory. However, you can also perform a batch publish on your own custom list of handler files.

Caution: The default batch publishing of all handlers should only be done when you install or upgrade CIC, unless you are instructed to do so by PureConnect Customer Care. In a batch publish of the default handlers, any customizations you may have made to your handlers (except [customization handlers](#)) are overwritten by new handlers with the same name.

If you are logged into the CIC Server as the CIC administrator or system administrator, or if you have rights in Interaction Administrator to publish handlers, you can perform a batch publish on the CIC server.

To publish a batch of default handlers:

1. Open a command prompt window on the CIC server.
2. Navigate to the directory that contains the handler [intermediate publish](#) (.i3pub) files for the default handlers. By default, these files are located on the CIC server in the directory `i3\ic\handlers\40Handlers`. The directory contains a .ihd and .i3pub file for each default handler.
3. Enter this command to create a file named `i3pubs.lst` in the current directory:

```
dir /b *.i3pub > i3pubs.lst
```

Note: You can optionally specify the activation status for each handler in the .lst file. For more information, see [format of the i3pubs.lst file](#).

4. Enter this command to publish the files listed in `i3pubs.lst`:

```
EICpublisherU @i3pubs.lst
```

For more information about this command, see [EICpublisherU](#).

To batch publish a custom list of handlers:

Follow these steps if you want to batch publish a list of custom handlers.

1. Open a command prompt window on the CIC server.
2. Change the current directory to the directory containing your custom handlers. By default, .ihd files for the custom handlers are located on the CIC server in the directory `i3\ic\handlers\custom`.
3. Use this command to create a text file that contains a list of the .ihd handler files in the current directory:

```
dir /b *.ihd > customhandlers.lst
```

4. Use this command to open Interaction Designer and publish the list of handlers:

```
idu /publish:customhandlers.lst
```

Note: To assign a category to a handler, you must [publish it from within Interaction Designer](#) or using the Interaction Designer COM API. For more information on using the Interaction Designer COM API, refer to the *Interaction Designer COM API* help in the PureConnect Documentation Library under the "System APIs" section.

Command Line Arguments for Batch Publishing

You may add the following command line arguments when performing a batch publish using the `idu` command. They should be used anytime you have an automated process doing a publish and you don't want Interaction Designer to stop the flow of things with modal message boxes.

/LogPublishEvents	Causes Interaction Designer to write messages to NT event log instead of popping modal dialogs. Note: If a handler contains an instance of the DB Stored Procedure tool, the /LogPublishEvents option will not prevent a dialog related to this tool from being displayed. This tool in particular may require DSN login information to be provided in order to work properly.
/ForcePublishClose	Causes Interaction Designer to close after batch publish even if there were errors.

.i3pub Files

A similar method can be employed for generating a number of [.i3pub files](#) at one time. At the command line, enter the following:

```
idu /intermediatepublish:{filename}
```

This works the same as the batch publish described above except that the files are not published. They are instead placed in the current directory as intermediate .i3pub files. The {filename} is the same format as for a batch publish.

Format of the i3pubs.lst File

The contents of the i3pubs.lst file that you create should look like the following:

```
HandlerOne.i3pub
HandlerTwo.i3pub
HandlerThree.i3pub
HandlerFour.i3pub
.
.
.
```

Optionally, you can specify the activation status ([primary](#) or [monitor](#)) for one or more handlers in the text file. This is especially useful when publishing custom handlers.

Consider this example:

```
FirstOne.i3pub
SecondOne.i3pub | Primary
ThirdOne.i3pub | Monitor
FourthOne.i3pub
.
.
.
```

In the example above, there will be no change to the activation status of FirstOne when published. If it had a Primary status before, it will remain primary. SecondOne will be set to Active Primary. ThirdOne will be set to Active Monitor. FourthOne will have no changes. If FirstOne or FourthOne had no activation status before (being published for the first time on the server), then the default status will be Inactive.

Activating and Deactivating Handlers

Active handlers (and subroutines) are published handlers that run on your CIC server. *Deactivated* handlers are published handlers that do not currently run on your CIC server. You can activate and deactivate handlers from Interaction Designer and Interaction Administrator. Handlers may be activated at the time they are published if desired, as described in the section on [Publishing a Handler or Subroutine](#). After publishing ([normal](#) or [batch publishing](#)), you may activate a previously inactive handler or deactivate an active handler. Both activating and deactivating handlers is performed in the [Manage Handlers notebook](#).

Note: This procedure describes how to activate and deactivate handlers from Interaction Designer. For information on activating and deactivating handlers from Interaction Administrator, see the Interaction Administrator online help.

To activate a handler from Interaction Designer:

This procedure may be performed after you have published your handler(s) and/or subroutine(s).

1. From the Interaction Designer Utilities menu, choose Manage Handlers.
The [Manage Handlers notebook](#) appears.
2. On the Primary Handlers page, select the handler(s) and/or subroutine(s) to activate in the Available Handlers list.

Note: If you don't see your handler listed in the Available Handlers list, check the Currently Selected Handlers list to see if your handler is already activated. If so, you do not need to activate the handler and can quit this procedure now. It may also be that the handler is running as a Monitor Handler. If the handler is running as a Currently Selected Handler on the monitor page, you do not need to activate it again and can quit this procedure now.

3. Click the Add button to move the selected handler(s) to the Currently Selected Handler list. When you click OK, the handler is activated and is available for use on the CIC server. You can now [debug](#) the handler if you want.

To deactivate a handler from Interaction Designer:

Note: You should perform this procedure if you want to stop a handler from running on the CIC Server.

Caution: If you deactivate a subroutine, and that subroutine is called by a currently running (activated) handler or subroutine, errors may occur.

1. From the Utilities menu, click Manage Handlers.
The [Manage Handlers notebook](#) appears.
2. On the Primary Handlers page, select the handler(s) and/or subroutine(s) you want to deactivate in the Currently Selected Handlers list.

Note: If you don't see your handler listed in the Currently Selected Handlers list, check the Available Handlers list to see if your handler is already deactivated. If so, you do not need to deactivate the handler and can quit this procedure now. It may also be that the handler is running as a Monitor Handler.

3. Click the Remove button to move the selected handler to the Available Handler list. When you click OK, the handler is deactivated and is no longer available for use on the CIC server.

Manage Handlers notebook

In the Manage Handlers notebook you can activate and deactivate Primary and Monitor handlers. All published handlers are listed here. Once you publish a handler, it appears in the Inactive Handlers list, unless it is already an actively running handler. CIC begins to use Activated handlers as soon as another event occurs for which the handler is registered. If Interaction Processor is currently running older versions of the handler, those threads finish before the new handler is used. Any threads running on handlers you deactivate continue until the thread is finished.

Caution: CIC cannot differentiate between primary and monitor handlers in the Manage Handlers Notebook. This is why the lists of inactive handlers are identical on the Primary Handlers and Monitor Handlers page. Make sure you don't activate a primary handler on the Monitor Handlers page, or vice versa.

Note: You can also manage handlers in Interaction Administrator's Server Configuration container.

Primary and Monitor Handlers

Primary handlers are generally handlers that act directly on objects such as calls within the system. Only one primary handler can be activated for a given initiator. For example, you cannot activate two primary handlers that both start with the Incoming Call initiator. Only one handler can act on the incoming call. This prevents two handlers from performing disparate actions on a single object. If you attempt to activate two handlers that start with the same initiator, CIC generates an error message in the event log.

Monitor handlers do not actively manipulate or modify objects in the system. Typically they retrieve call attributes and write that information to a database for reporting purposes, although they are not limited to reporting. CustomCallDisconnectMonitor determines if a call was recorded, and if so, where to send a copy of that recording. Since monitor handlers are not acting on objects, more than one monitor handler can use the same initiator. For example, CallDisconnectMonitor and CustomCallDisconnectMonitor both use the Call Monitor Initiator configured to start when a call disconnects.

Activating and Deactivating Handlers

Click on one of the links below for more information on activating primary and monitor handlers.

[Primary Handlers page](#)

[Monitor Handlers page](#)

View Dependencies

When developing handlers, it is sometimes useful to see how all the parts fit together. A common question is, "What handlers call a certain subroutine?" The Dependency Viewer allows you to see the interaction between the objects that make up handlers and the handler themselves.

Interaction Designer supports the ability to track dependencies for the following objects used in handlers:

- DB Profiles
- Global Variables
- HTML Templates
- Initiators
- InternationalString Resources
- Prompt Resources
- Sequence Resources
- Subroutines
- Tools

Choose the dependency that you want to view by selecting the appropriate item from the Dependency listbox. Click the plus signs (+) beside an object shows its contents. Users can choose to group each type of dependency information in two ways. You can either select to view the dependency information by what items (variables, tools, etc.) are used by a handler, or by handlers that use each item by clicking on the appropriate "Group By" radio button on the dialog.

One thing to remember is that the dependency information displayed in this dialog is for published handlers only. Actions performed on a handler that has not been published will not cause an update in the dependency information.

The Dependency View offers the following three options.

Export Dependencies to XML

This exports the selected dependency information as an XML document. The user is prompted to supply the location for this XML document.

[Click here](#) for more information on exporting dependencies to XML.

Download Handler

This downloads the selected handler to the directory specified in the [Designer Preferences](#) page. If no handler is specified there, the user is prompted to supply a destination for the download.

Debug Handler

This opens the selected handler and initiates a debugging session.

Views Preferences

This tab in the Designer Preferences dialog box contains options you can set to affect the initial appearance of the Design window each time a handler opens.

To display the Designer Preferences dialog box, click the Edit menu, click Preferences, and then click Designer Preferences.

Zoom

Set the initial zoom of the Design Window. This initial zoom can be anything from 10% to 200%. If a zoom size is set that is not one of the standard sizes, that size will be added to the Zoom options given under the Layout menu.

Maximize on open

Select this checkbox if you want to have every handler automatically maximize on open.

Related Topics

[Debugging Preferences](#)

[Designer Preferences Page](#)

[General Preferences](#)

[Handler Download Preferences](#)

Customizing Handlers

Customization Points

Several handlers contain customization points. These are places where you can implement customizations and not worry about future CIC releases overwriting your customizations. Customization points are just calls to customization subroutines. These customization subroutine calls will always be in the same location, so there is no danger of future hotfixes or service releases overwriting them.

Warning: Do not make changes in any base handler outside of the designated customization points. Doing so may cause CIC to stop functioning properly. If you feel you must make a change to a handler outside of the designated customization points, contact PureConnect Customer Care.

Customization subroutines can be located by using the [Dependency Viewer](#). View the subroutine dependencies; all of the customization subroutines have a naming scheme of "Custom" followed by an indicator of what it is intended to customize. For example, "CustomHeldCallTimer.ihd" is the customization point provided for custom processing on calls that have remained on hold longer than configured in the Held Call Timer server parameter.

Default System Event Handlers

Following is a list of the system event handlers that ship with IC 3.0.

Warning: These handlers should never be modified in any way. Customization points have been placed within these handlers that call custom subroutines. Any custom processing you want to add to these handlers should be done in those custom subroutines and not in the handlers themselves. Any modifications made directly to these base handlers may cause your system to become unstable.

System_AsynchronousDigitsReceived

This handler starts any time a GetDigitsExAsynch event is created by the [Extended Get Key Async](#) tool. This handler picks up call processing where it was left off in the handler that generated the event.

System_CallOfferingNonSystemQueue

This handler determines the type of queue (user or workgroup) that a call should be routed to and calls the appropriate subroutine.

System_CallOfferingOperatorAttendant

This handler starts any time a Transfer to the operator is generated. The event is generated by handlers when a call on the IVR or a call in the Interaction Attendant transfers to the operator.

System_CallOfferingOutboundAttendant

This handler starts any time a Transfer to the Outbound Attendant is generated. The event is generated by handlers when a call on the IVR or a call in the Interaction Attendant transfers to the Outbound Attendant. Calls from Interaction Dialer may generate this event when sending calls to an IVR.

System_CallOfferingSystemQueue

This handler starts any time a Transfer to System Queue event is generated. This event is generated any time a call is placed on the system queue. The SetCallState step changes the state to connected, but it will fail if the call object no longer exists (i.e. the call disconnected after this handler started). This handler is called only when a call is sent to the System queue after some other IVR processing has been performed. New incoming calls are placed on the System queue by the Sytem_IncomingCall handler.

System_ClientDisconnect

This handler is intended to allow users to create custom handling for telephone calls being disconnected. This handler runs whenever a Custom Client Disconnect initiator is fired. Once initiated, the handler queries the media type for the interaction to be disconnected. If the interaction is a telephone call, the CustomClientDisconnect subroutine is called, then the call is disconnected. For all other media, the interaction is simply disconnected.

System_ClientPlayRecording

This handler starts any time the client plays a recording through a user's currently logged in station.

System_ClientPromptRequest

This handler starts when a CIC client user presses one of the record prompt buttons located on the Configuration page. The three types of prompts that can be recorded are Name, No Answer, Out of Office, and Smile. Information about the station and user queue from which the request originated is also collected.

System_ContinuousListenRequest

This handler processes continuous monitoring requests from a CIC client. It simply calls SystemContinuousListen to perform the monitor function.

System_EmailOfferingInteractionAttendant

This handler starts any time an email interaction is transferred from one node to another in the Email Attendant. This event is generated by handlers.

System_FaxSendCompleted

This handler runs after a fax was successfully or unsuccessfully sent. After this handler runs, the server copy of the fax file is deleted. If the Fax send was successful, an email is sent to the sender indicating success. If the fax send failed, an email indicating the failure is sent to the sender. If the fax send failed, and the fax file was successfully created, the fax file is included in the failed message email to the sender so that he or she can try resending.

System_GenericObjectOfferingNonSystemQueue

This handler uses the Query Queue Type tool to determine if the queue id is a user or workgroup queue and will send the call into SystemIVRUserQueueCallback or SystemIVRWorkgroupQueue as appropriate.

System_HeldInteractionTimer

This handler starts when a call is on hold for a period of time longer than is specified in the Held Interaction Timeout server parameter in Interaction Administrator. The handler then calls CustomHeldCallTimer to perform special processing on that call. There is no processing by default, but you could configure anything you like.

System_IncomingCall

This handler answers an incoming call when it first comes in to CIC. System_Incoming Call then calls several customization points to process the new call. This handler then calls SystemIVR where prompts are played and the caller's key presses are evaluated.

System_IncomingFax

System_IncomingFax looks for information in the Fax object and the IncomingFax event and uses this information to route the fax to a recipient. This handler sends the fax file as an attachment to an email. It runs after the Fax server finishes receiving a fax.

System_IncomingInteraction

This handler processes an incoming web interaction when it first enters CIC and transfers it to the appropriate queue.

System_IncomingQueueEmail

This handler is initiated by the postmaster server any time an incoming email is detected. The handler retrieves the attributes of the email object, formats the body and subject of the email, assigns any new attributes, and transfers the email object to the appropriate queue.

System_IncomingSMS

This handler processes NewIncomingSMS events. It first calls CustomIncomingSMS, so that any customization can happen before the standard process. If the handler is allowed to continue after the customization point, it fetches the System Parameter "SMS Inbox" for the name of the mailbox the SMS should be sent to. If the System Parameter does not exist, the SMSObject is simply disconnected.

In the other case, it queries the System Parameter "SMS Outbox", and will use the value for the ReplyTo field of the email. This allows the agent or the user to reply to an SMS Message with another SMS Message. When the email is sent the SMSObject is disconnected.

- The *Subject* will contain the phone number used to send the SMS Message to the PureConnect Platform.
- The *Body* will contain the text of the SMS Message.
- *Localization Note*—the Template String SYSTEM_INCOMING_SMS_SUBJECT is used for the text of the Subject.

System_IncomingSR

This handler processes NewIncomingSR events. It first calls CustomIncomingSR, so that any customization can happen before the standard process. If the handler is allowed to continue after the customization point, it fetches the System Parameter "SMS Inbox" for the name of the mailbox the SMS should be sent to. If the System Parameter does not exist, the SMSObject is simply disconnected. When the email is sent the SMSObject is disconnected.

- The *Subject* will contain the phone number used to send the SMS Status Report to the PureConnect Platform.
- The *Body* will contain the TicketId, the SMS Status, the SMS Detailed Status, the Client Id, and the Broker Date of the SMS Status Report.
- *Localization Note*—the Template String SYSTEM_INCOMING_SR_SUBJECT is used for the text of the Subject. The Template String SYSTEM_INCOMING_SR_BODY is used for the text of the Body.

System_InitiateCallRequest

This handler places an intercom call or a call on an outside line. This handler is started when someone places a call using a CIC client or places a call from a station.

System_InitiatePage

This handler is used to execute a page through the client.

System_InteractionOfferingNonSystemQueue

This handler launches in response to an Interaction on Non System Queue event. It first determines the type of interaction, then routes the interaction to the appropriate queue.

System_InteractionVoicemail

This handler sends a chat session or callback to voicemail.

System_MessageLight

This handler starts when a voicemail is left, a message is sent from remote voicemail, or when a reply is made to a previously left voicemail. This handler will check each station associated with the message and set the voicemail indicator to on or off, depending on whether or not there are unchecked messages still at that station.

System_OutgoingFax

This handler runs when a fax is sent manually from the Interaction Fax viewer. The only action this handler performs is to queue the fax for sending.

System_OutgoingQueueEmail

This handler prepares an outgoing email to be sent. Any necessary formatting such as adding a "Re:" prefix to the subject line is done, any attachments are bundled together, and the email object is transferred to the outgoing mail queue.

System_OutgoingSMS

This handler processes NewOutgoingSMS events. It is usually triggered from a Client application or via the SMS Send tool from a handler. It just calls the SMS CompleteSend tool to instruct the SMSServer to actually send the SMS Message to the SMS Broker. It simply disconnects the SMSObject on success. On failure, it will try to get the SMS Results from the SMSObject and calls CustomSystemError, then disconnects the SMSObject.

System_OutgoingSMSEmail.ihd

This handler monitors a mailbox and will use its incoming emails to send SMS Messages. The Subject must contain the destination(s). In case of more than one destination, they must be separated with a comma, a semi-colon, a pipe, or a forward slash. The body of the email is the text of the SMS Message. If the subject does not resolve in a list of valid phone numbers (according to your Dial Plan), the process is cancelled silently.

Note: The shipped handler monitors the mailbox number 12. It is the responsibility of the CIC Administrator to set up that mailbox in Interaction Administrator. When this operation is done the first time, it is advisable to reopen this handler and verify that it actually points to mailbox number 12.

System_ParallelMakeCallProcess

This handler initiates multiple call threads in the follow me functionality.

System_QueueEmailOfferingNonSystemQueue

This handler starts any time a Transfer, to a user or workgroup, of a non-ACD Queue Routed Email is generated. For example, the event will fire when an agent transfers an email interaction in the client to another user or workgroup. This handler determines the type of queue (user or workgroup) that an email is routed to and calls the appropriate subroutine.

System_StationOffHook

This handler starts when someone picks up a station. This handler determines if there is a call waiting to be picked up. If there is a call waiting to be picked up, that call is connected. If there is not a call waiting to be picked up, dialtone is played and the caller can dial a number. When dialing, the caller may also enter Account and/or Authorization codes if necessary before dialing. The Station Place Call then generates an Outgoing Call Request event that will start the System_InitiateCallRequest handler.

System_SupervisorAlert

This handler starts any time Interaction Supervisor sends an email alert. Interaction Supervisor alert data is formatted and packed in an email message and sent to the intended recipient.

System_SwitchhookFlash

This handler starts when a user causes a flash event from a station connected to CIC. Flashes can be created either by quickly depressing and releasing the station switch hook, or by pressing the telephone's flash button (if one exists). The purpose of this handler is to allow station users (who are not running a CIC client) to perform basic operations such as transferring, conferencing, and checking the contents of a station queue.

System_TransferCallRequest

This handler transfers a call to an internal extension or outside telephone number. This handler starts when someone transfers a call.

System_TransferToVoicemailRequest

This handler sets a call's state to 'Voice Mail' and calls the VoiceMail subroutine. It is called from handlers, and when a CIC client user clicks the Voice Mail button. This handler also identifies the type of queue the call is on, and plays the appropriate prompts for that queue's voicemail. If the caller is still connected after leaving voicemail, the call is transferred to the system queue.

System_TUIMenu

This handler starts any time the TUI is initiated in asynchronous mode. In asynchronous mode the handlers send an event to start a new handler thread for processing the TUI. This allows for freeing the prior handler call stack so memory can be freed before entering the TUI.

System_VoxFormRequest

This handler starts any time the voicemail player plays a recording through a user's currently logged in station.

System_WebCallBack

This handler is initiated from a callback request made at a website. This handler routes the request and sends an email to the recipient containing information pertinent to the request and requestor.

System_WrapUpRequest

This handler starts any time an ACD call or a non-ACD call to a workgroup is flagged for an agent to enter a wrap-up code for the call. The event is generated by handlers when a call flagged for wrap-up completes.

Retrieving the values of server and system parameters from handlers

Server and system parameters are variables whose values are set in Interaction Administrator. These values can be retrieved from handlers, and from other locations within Interaction Administrator. For example, the Held Call Timeout server parameter determines the number of seconds a call remains on hold before the Held Call Timer initiator starts a handler.

You can create your own custom server and system parameters and retrieve their values from within handlers. For example, suppose you want to create a handler that retrieves the email address of the web administrator. If you hard coded the email address in Send Email steps, you would have to modify your handlers each time you change web administrators. If the email address is stored in a server parameter, any handler can retrieve that value each time it runs. When you change the email address stored in the server parameter, all the handlers that retrieve that server parameter value are dynamically updated the next time they run.

See the Interaction Administrator online help for a list of default server and system parameters.

The difference between server and system parameters

Server parameters are available only on a particular CIC server and system parameters are available on all CIC servers on a network. In a future release of CIC that supports multiple CIC servers, system parameters will become more useful. Until that time, you can store values in either server or system parameters.

Caution: Any custom server or system parameters you create are overwritten if you perform a full install. If you are just performing an upgrade or refresh install, you will not lose your custom server and system parameters.

To create a server or system parameter:

You can create, edit, and delete server and system parameters in Interaction Administrator. See the Interaction Administrator documentation for complete instructions on configuring server and system parameters.

After you have created (and assigned a value to) a server or system parameter, you can retrieve that value from a handler.

To retrieve the value of a server or system parameter from a handler:

Server and system parameter values are retrieved with a [GetDSAttr](#) step. The GetDSAttr step requires two input fields, *Directory Services Path* and *Directory Services Attribute*. Use the following descriptions as guidelines for the values you should place in these fields when you retrieve a custom server or system parameter value.

Directory Services Path

This is the path to server or system parameter in Directory Services. In the following examples, substitute `your_server_parameter_name` with the actual name of your server or system parameter.

If you created a server parameter, type:

```
"${Server}\Parameters\your_server_parameter_name"
```

If you created a system parameter, type:

```
"${Config}\Parameters\your_system_parameter_name"
```

Directory Services Attribute

This is the name of the attribute you want to retrieve. For custom server and system parameters, you should type:

"Value"

Specify an output variable to contain the List of Attribute Value. This output will contain the value in the custom server or system parameter. Remember that this is a List of String type variable. If you need to convert the value to a string, use the [GetHead](#) operation.

Once you have saved and published the handler, you can begin retrieving the values of system and server parameters from within handlers.

Attributes that can be looked up in Directory Services Keys

An attribute is a piece of information about a Directory Services key. For example, some attributes of a User Key include extension, user queue identifier, and e-mail address (Mailbox). This topic lists many of the attributes that can be looked up in the Directory

Service keys: Users, Lines, Line Groups, Workgroups, Stations. This is a long topic, and you may want to print it to keep as a reference.

Notes: Don't confuse attributes found in a Directory Services key with object attributes, which are pieces of information about a call or chat object that travel with that object. For more information, refer to the Interaction Attributes Reference Guide (attrib.chm) in the System APIs section of the PureConnect Documentation Library.

Attributes that start with a lowercase letter are set in e-mail server administration application (such as MS Exchange Administrator or IBM Domino Administrator). Those that start with an uppercase letter are set in Interaction Administrator (IA).

User Key

Attribute Name	Description	Example String	Set from...
ACD Agent Greeting	Set to Yes if the Agent Greeting "Enable" checkbox is selected.	"Yes"	IA User configuration > ACD tab, Options page
ACD Agent Greeting File	File name of the Agent Greeting prompt.	agent1.wav	IA User configuration > ACD tab, Options page
Account Code Required	Set to yes if the role (e.g., Business User, Agent, etc.) has the "Account Verification" option selected.	"Yes"	IA configuration of Default User, User, Role, or Workgroup, Security tab, and User Rights options
Account Codes List	Lists the account code(s) this user, role, or workgroup can choose.	"01 02 03" or "[[All]]"	IA configuration of default User, User Role, or Workgroup, Access Control tab, and select the Account Codes category
ACL Admin	Permission for user to view Admin Access page in IA.	"Yes"	IA User Configuration> Admin Access>Show Access Control Page
Admin ACL Admin	Permission for user to view Access Control page in IA.	"No"	IA User Configuration > Admin Access>Show Admin Control Page
Allow Email	Set to Yes if the "Allow E-mail Access via TUI" option is selected.	"Yes"	IA configuration of Default User, User, Role, or Workgroup, Security tab, and User Rights options
Allow Fax	Set to Yes if the "Allow Fax Access via TUI" option is selected.	"Yes"	IA configuration of Default User, User, Role, or Workgroup, Security tab, and User Rights options
Allow Voice Mail	Set to Yes if the "Allow Voice Mail Access via TUI" option is selected.	"Yes"	IA configuration of Default User, User, Role, or Workgroup, Security tab, and User Rights options
Auto-Answer Call	Automatically connect ACD calls to this user's phone.	"Yes"	IA User Configuration > ACD
Auto-Answer Non-ACD Calls	Set to Yes if the "Auto-Answer non-ACD Interactions" option is selected.	"Yes"	IA User Configuration > ACD tab
Available Forward Message	File name of the user's Available Forward prompt.	AFPrompt_agent1.wav	CIC client > Configuration > Personal Prompts page > Record Available, Forward Message button
Call Fwd All Mode	Indicates a user's type of Call Coverage "Forward" option.	"All" "Internal" "External" "Unknown"	CIC client > Configuration > Call Coverage page > Coverage Options, Forward checkbox and list

Call Fwd Phone Number	Indicates the forward extension when a user has "Call Coverage" forwarding rules set.	"8113", "2501", etc.	CIC client > Configuration > Call Coverage page > Coverage Number, "Send my calls to" field
Call Waiting	Indicates if the user has enabled call waiting.	"Yes"	CIC client > Configuration > Calls page > "Enable call waiting" checkbox
Classification List	Contains the list of phone number classifications assigned to a role, workgroup, or user.	"Toll Free Intercom Local Emergency"	IA configuration of Default User, User, Role, or Workgroup, Access Control tab, and Phone Number - Classifications category
CompanyName	Company name.	"Interactive Intelligence, Inc."	Email Server Administrator program
Customize Client	Allows user to customize the settings on their client workstation.	"No"	IA User Configuration > Basic Security
Date Created	The date on which this user key was created.	"908400570"	IA User Configuration > History
Date Last Modified	The date on which an attribute in this user key was last modified.	"913150263"	IA User Configuration > History
Default Workstation	From the Default Workstation list, select the name of the workstation primarily associated with the user's account.	"STEPHENSPC"	IA User Configuration > Configuration
DefaultMailFolder	User's default mail folder for accessing e-mail, voice mail, and faxes.	"Inbox"	
DisplayName	Name displayed in company directory and other locations.	"Schiller, Stephen"	Email Server Administrator program, mailbox properties
EmailAddress	Exchange email address (cannot be changed).	/o=i3/ou=i3-Home/cn=Recipient/cn=StephensS	Email Server Administrator program
EmailAlias	Email alias.	"StephensS"	Email Server Administrator program
EmsExtensionAttribute1	Additional exchange attributes.	"Jan 29"	Email Server Administrator program
EmsExtensionAttribute2	Additional exchange attributes.	"stephens@ibm.net"	Email Server Administrator program
EmsProxyAddresses	Set from Exchange. Do not modify.		Email Server Administrator program
EmsUSNChanged	Set from Exchange. Do not modify.		Email Server Administrator program
Exclude From Directory	Set to Yes if this option is selected. Indicates if user is listed in Company Directory.	"Yes"	IA User Configuration ? Configuration tab > Exclude from directory check box
Extended Absence Message	Indicates the prompt file name if the user recorded an "Out of office message."	\\3\IC\Resources\EAPrompt_Abell.wav	CIC client > Configuration > Personal Prompts page > Record out of office message button

Extended Absence Prompt	Set to Yes if the user recorded an "Out of office message."	"Yes"	CIC client > Configuration > Personal Prompts page > Record out of office message button
Extension	A unique (logical) extension number associated with this user.	"114"	IA User Configuration > Configuration
Fax Capability	Set to Yes if user or workgroup is enabled to receive faxes.	"Yes"	IA User Configuration > Options > Fax Capability checkbox
Fax Use TIFF	Set to yes if user, role, or workgroup is enabled to receive faxes.	"Yes"	IA User, Workgroup or Role Configuration > Security tab > User Rights 2 page > Use TIFF for faxes checkbox
Follow Me	Set to yes if user has rights to use "Available, Forward" status.	"Yes"	IA User Configuration > Security tab > User Rights page > Follow Me checkbox
Follow Me Alert Timeout	Indicates the timeout (seconds) a follow-me call alerts before trying next number.	15 30	CIC client > Configuration > Follow Me page
Follow Me Message	Indicates the Follow-me message prompt name.	D:\I3\IC\Resources\FMPrompt_Abell.wav	CIC client > Configuration > Personal Prompts page > Record follow-me message button
Follow Me Pin Authentication	Indicates if the user is required to enter CIC password PIN to receive forwarded calls.	"Yes Yes"	CIC client > Configuration > Follow Me page
Follow Me Telephone Numbers	Indicates the user's follow-me phone number(s).	7654699048 3174743633	CIC client > Configuration > Follow Me page
Focus	Gives the CIC client focus when a call alerts.	"Yes"	IA User Configuration > Client Notification
Givenname	First name.	"Allen"	Email server Administrator program
Home Site	The home site number for the user.	111	IA User Configuration > Configuration tab
Listen In	Permission to use the listen in button on the CIC client	"No"	IA User Configuration > Basic Security
Mailbox	The mailbox into which voicemail for this user is placed.	"EX:/0=i3/ou=i3-Home/cn=Recipient/cn=Stephens"	IA User Configuration > Configuration
MWI Address	A user has MWI enabled and configured to notify a specific address.	"1234"	IA User Configuration > MWI
MWI Enabled	Indicates if the user has MWI enabled.	"Yes"	IA User Configuration > MWI
MWI Mode	Indicates user's MWI status.	"0" - Use default or logged in workstation "1" - Send to SDMI port "2" - Send to MWI Address	IA User Configuration > MWI
Manager Handlers	Permission to activate and deactivate handlers.	"Yes"	IA User Configuration > Basic Security

Master Admin	Top level permissions in IA.	"Yes"	IA User Configuration > Admin Access
Message Notification Rights	An alerting configuration option that enables users to modify the general and parked call alerting options.	"Yes"	IA Client Configuration Template > Alerting page
Modify Attendant Configurations	Permission to modify Interaction Attendant configurations in IA.	[[All]] "Inbound" "Outbound"	IA User, Default User, or Role Configuration > Access Control > Modify Attendant Configurations
Modify Line Queue List	Permission to modify a Line Queue list in IA.	"LineQueue1" "LineQueue2"	IA User Configuration > Access Control > Modify Line Queue
Modify Station Queue List	Permission to modify a Station Queue list in IA.	[[All]]"	IA User Configuration > Access Control > Modify Station Queue
Modify User Queue List	Permission to modify a User Queue list in IA.	[[All]Administration]" "KevinK" "StephenS"	IA User Configuration > Access Control > Modify User Queue
Modify Workgroup Queue List	Permission to modify a Workgroup Queue list in IA.	"Marketing" "Development"	IA User Configuration > Access Control > Modify Workgroup Queue
Name Prompt	Set when a user records a name prompt.		CIC client > Configuration
Name Prompt Reminder	Set by a handler to monitor the status of sending a user an e-mail reminder about recording their name.	"Sent"	Handlers
Name TTS Spelling	A phonetic spelling of a user's name to help the TTS engine "say" it accurately.	"Kevin Koonz"	IA User Configuration ? Phonetic Spellings
No Answer Message	Set when a user records a No Answer Message.		CIC client > Configuration
Notes	Notes typed on History page in IA.	"This is a note."	IA User Configuration > History
NT Domain User	Windows NT domain user name.	"domain\user"	IA User Configuration > Configuration
Offering Call Timeout	The number of seconds a call will alert before a handler takes over and performs special processing, such as Vmail.	"0"	IA Default User Configuration > Options
Operator Target Number	The number to dial for an operator.	"0"	Interaction Attendant
Parked Call Extension	The destination extension for a parked call that has reached its timeout. Used in conjunction with Parked Call Timeout.		IA Default User Configuration > Options
PhonePrimaryFax	Fax number	"3177158114"	Email Server Administrator program

Phone Remote	The phone number to use with the Available Forward status to forward calls to a remote number.	"7657290931"	CIC client > Configuration
PhoneHome1	The user's primary home telephone number.	"7657163861"	IA > Personal Info > General page
PhoneMobile	The user's mobile phone number.	"3177280352"	IA > Personal Info > More Phone Numbers page
Pick Up Held Call	Indicates if the user has enabled the option to pick up the oldest held call when picking up the phone, unless there is an alerting call.	"Yes"	CIC client > Configuration > Calls page > Connect to Oldest Held Call When I Pick up the Phone
Pop Client	Opens the CIC client when a call alerts.	"Yes"	IA User Configuration > Client Notification
PostalCode	Postal code (ZIP code).	"46268"	Email Server Administrator program
Private	Permission to set a call as private.	"Yes"	IA User Configuration > Basic Security
Publish Handlers	Permission to publish handlers.	"Yes"	IA User Configuration > Basic Security
Queue Identifier	Looking up this attribute will return the fully qualified queue identifier; however, this attribute does not actually exist in the Windows NT registry. The formula for this string is "User Queue:" + the CIC username.	"User Queue:StephenS"	(Internal)
Record	Permission to record a call.	"Yes"	IA User Configuration > Basic Security
Require Forced Authorization Code	Indicates whether or not a user's station phone allows toll calls without an authorization code.	"Yes"	IA User Configuration > Security Rights > User Rights 2
Ring Computer	Audio alert through PC for incoming calls.	"Yes"	IA User Configuration > Client Notification
Ring Phone	Telephone rings for incoming calls.	"Yes"	IA User Configuration > Client Notification
Screen Calls	Requires callers to record their name so it can be displayed to the agent before the agent accepts the call.	"Yes"	IA Client Configuration Template > Follow Me page
Show Config Page	Permission to display Configuration page in CIC client.	"Yes"	IA User Configuration > Basic Security
Show Lines Queue Page	Permission to display Lines Queue page in CIC client.	"Yes"	IA User Configuration > Basic Security

Show Report Page	Permission to display Report page in CIC client.	"Yes"	IA User Configuration > Basic Security
Show Stations Queue Page	Permission to display Stations Queue page in CIC client.	"Yes"	IA User Configuration > Basic Security
Show System Queue Page	Permission to display System Queue page in CIC client.	"Yes"	IA User Configuration > Basic Security
Show Users Queue Page	Permission to display Users Queue page in CIC client.	"Yes"	IA User Configuration > Basic Security
Show Workgroups Page	Permission to display Workgroups page in CIC client.	"Yes"	IA User Configuration > Basic Security
Show Workgroups Queue Page	Permission to display Workgroups Queue page in CIC client.	"Yes"	IA User Configuration > Basic Security
Skills	The skills associated with a workgroup in the form of skill name, proficiency, and desire to use.	"Spanish 3 4 "	IA Skill > Configuration
SnCreated	For system use only. Do not modify these values.	_____	_____
SnModified	For system use only. Do not modify these values.	_____	_____
StateOrProvince	State or province.	"IN"	Email Server Administrator program
Status DND	Indicates whether the user's current status is a DND status.	"Yes"	CIC client interactions page.
Status Text	The text of the user's status.	"Do Not Disturb"	CIC client interactions page.
Status Until Date	If set, the date the user has set to return.	"1998-12-7 00:00:00"	CIC client interactions page.
Status Until Time	If set, the time the user has set to return.	"10:00:00 AM"	CIC client interactions page.
StreetAddress	Street address	"1234 Abby Rd."	Email Server Administrator program
surName	Last name	"Schiller"	Email Server Administrator program
Title	Title	"Technical Communicator"	Email Server Administrator program

View Line Queue List	List of Line Queues this user can view in CIC client.	"LineQueue1" "LineQueue2"	IA User Configuration> Access Control
View Reports List	List of Reports this user can view in CIC client.	"[[All]]" "[Graph] Area Code"	IA User Configuration> Access Control
View Station Queue List	List of Station Queues this user can view in CIC client.	"[[All]]" "Line1Phone" "Fax1"	IA User Configuration> Access Control
View User Queue List	List of User Queues this user can view in CIC client.	"[[All]]Administration" "KevinK" "StephenS"	IA User Configuration> Access Control
View Workgroup List	List of Workgroup Queues this user can view in CIC client.	"Marketing" "Development"	IA User Configuration> Access Control
View Workgroup Queue List	List of Line Queues this user can view in CIC client.	"Administration" "Support"	IA User Configuration> Access Control
Whisper Tone Level	Slider bar setting that indicates the volume of the whisper tone an agent hears when there is an ACD call to the workgroup.		IA Default User Configuration > ACD Options
Workgroups	List of Workgroups of which user is a member.	"Marketing" "Support"	IA User Configuration> Workgroups

Workgroup

Attribute Name	Description	Example	Set from...
900 Service	Permission to call 900 numbers.	"No"	IA Workgroup Configuration > Basic Security
ACD Execute Transfer On User Transfer	Invokes Transfer Action DDE command.	"No"	IA Workgroup Configuration > Actions
ACL Admin	Permission for members to view Admin Access page in IA.	"Yes"	IA Workgroup Configuration > Admin Access
Active	Indicates if workgroup is active or inactive.	"Yes"	IA Workgroup Configuration > Configuration
Admin ACL Admin	Permission for members to view Access Control page in IA.	"Yes"	IA Workgroup Configuration > Admin Access
ACD Offering Action	DDE action.		
Customize Client	Permission to customize CIC client.	"Yes"	IA Workgroup Configuration > Basic Security
Date Created	The date on which this workgroup key was created.	"908400570"	IA Workgroup Configuration > History

Date Last Modified	The date on which an attribute in this workgroup key was last modified.	"913150263"	IA Workgroup Configuration > History
EmailAddress	The mailbox into which voicemail for this workgroup is placed.	"913150263"	IA Workgroup Configuration > History
Extension	A unique (logical) extension number associated with this workgroup.	"3"	IA Workgroup Configuration > Configuration
Has Queue	Provides a queue into which calls can be placed.	"Yes"	IA Workgroup Configuration > Configuration
International	Permission to call international numbers.	"Yes"	IA Workgroup Configuration > Basic Security
Listen In	Permission to use the Listen In button in CIC client.	"Yes"	IA Workgroup Configuration > Basic Security
Long Distance	Permission to call long distance numbers.	"Yes"	IA Workgroup Configuration > Basic Security
Manager Handlers	Permission to activate and deactivate handlers.	"Yes"	IA Workgroup Configuration > Basic Security
Master Admin	Top level permissions in IA.	"Yes"	IA User Configuration > Admin Access
Modify Workgroup Queue List	Permission to modify a Workgroup Queue list in IA.	"Marketing" "Development"	IA User Configuration > Access Control > Modify Workgroup Queue
Notes	Notes typed on History page in IA.	"This is a note."	IA Workgroup Configuration > History
Offering Call Timeout	The number of seconds a call will alert before a handler takes over and performs special processing, such as voicemail.	"0"	IA Workgroup Configuration > Options
Private	Permission to set a call as private.	"Yes"	IA Workgroup Configuration > Basic Security
Publish Handlers	Permission to publish handlers.	"Yes"	IA Workgroup Configuration > Basic Security
Queue Identifier	Looking up this attribute will return the fully qualified queue identifier; however, this attribute does not actually exist in the Windows NT registry. The formula for this string is "Workgroup Queue:" + the CIC workgroup name.	"Workgroup Queue:Marketing"	(Internal)
Record	Permission to record a call.	"Yes"	IA Workgroup Configuration > Basic Security
Show Config Page	Permission to display Configuration page in CIC client.	"Yes"	IA User Configuration > Basic Security
Show Lines Queue Page	Permission to display Lines Queue page in CIC client.	"Yes"	IA Workgroup Configuration > Basic Security

Show Report Page	Permission to display Report page in CIC client.	"Yes"	IA Workgroup Configuration > Basic Security
Show Stations Queue Page	Permission to display Stations Queue page in CIC client.	"Yes"	IA Workgroup Configuration > Basic Security
Show System Queue Page	Permission to display System Queue page in CIC client.	"Yes"	IA Workgroup Configuration > Basic Security
Show Users Queue Page	Permission to display Users Queue page in CIC client.	"Yes"	IA Workgroup Configuration > Basic Security
Show Workgroups Page	Permission to display Workgroups page in CIC client.	"Yes"	IA Workgroup Configuration > Basic Security
Show Workgroups Queue Page	Permission to display Workgroups Queue page in CIC client.	"Yes"	IA Workgroup Configuration > Basic Security
View Workgroup Queue List	List of Workgroup Queues members of this workgroup can view in CIC client.	"Marketing" "Development"	IA Workgroup Configuration > Access Control
snCreated	For system use only. Do not modify these values.	_____	_____
snModified	For system use only. Do not modify these values.	_____	_____
WorkGroup icon	Icon displayed in CIC client	"c:\test.ico"	IA Workgroup Configuration > Configuration page
Workgroup Ring Sound	Sound played when a call alerts.	"c:\test.wav"	IA Workgroup Configuration > Configuration page
Wrapup Status	Status displayed during wrapup time.	"Wrapping Up"	IA Workgroup Configuration > Actions page
Wrapup Time	The number of seconds to allow a CIC client user to finish any work associated with the previous call before becoming available to receive a new call.	"5"	IA Workgroup Configuration > Actions page

Workstation

Attribute Name	Description	Example	Set from...
900 Service	Permission to call 900 numbers.	"No"	IA Station Configuration > Workstation Configuration
Active	Indicates if station is active or inactive.	"Yes"	IA Station Configuration > Workstation Configuration
Board Number	Type the number of the Dialogic station device interface board supporting this workstation.	"9"	IA Station Configuration > Workstation Configuration
Channel	SCBus Fax device channel number (only present if SCBus type fax device).	"1"	
Date Created	The date on which this station key was created.	"908400570"	IA Station Configuration > History
Date Last Modified	The date on which an attribute in this station key was last modified.	"913150263"	IA Station Configuration > History
Driver			
Extension	A unique (logical) extension number associated with this workstation.	"5"	IA Station Configuration > Workstation Configuration
International	Permission to call international numbers.	"No"	IA Station Configuration > Workstation Configuration
Long Distance	Permission to call long distance numbers.	"No"	IA Station Configuration > Workstation Configuration
Notes	Notes typed on History page in IA.	"This is a note."	IA Station Configuration > History
Port Number	The port number for this workstation.	"9"	IA Station Configuration > Workstation Configuration
Queue Identifier	Looking up this attribute will return the fully qualified queue identifier; however, this attribute does not actually exist in the Windows NT registry. The formula for this string is "Station Queue:" + the CIC station name.	"Station Queue:STEPHENSPC"	(Internal)
Remote Phone Number			
Ring Always	Determines whether station will ring for alerting calls.	"Yes"	IA Station Configuration > Workstation Configuration
SnCreated	For system use only. Do not modify these values.	_____	_____
snModified	For system use only. Do not modify these values.	_____	_____
Station Type	Station type, i.e. workstation, stand alone phone, analog fax device, SCBus fax device, etc.	"Workstation"	IA Station Configuration > Workstation Configuration

Line Key

Attribute Name	Description	Example	Set from...
Active	Indicates if line is active.	"Yes"	IA Line Configuration > Configuration
Board Number	The voice board number associated with this analog line.	"1"	IA Line Configuration > Configuration
Board Type	Line card type.	"Analog"	IA > Insert new line wizard
Caller ID Enabled	Determines is caller ID is enabled for this line.	"Yes"	IA Line Configuration > Caller ID
Caller ID Format	One of three caller ID formats.	"Class" or "ACLIP" or "CLIP"	IA Line Configuration > Caller ID
Date Created	The date on which this line key was created.	"908400570"	IA Line Configuration > History
Date Last Modified	The date on which an attribute in this line key was last modified.	"913150263"	IA Line Configuration > History
Direction	Determines whether line accepts inbound, outbound, or both types of calls.	"Inbound" or "Outbound" or "Both"	IA Line Configuration > Configuration
DNIS Enabled	Determines whether line is DNIS enabled.	"Yes"	IA Line Configuration > Caller ID
Inband Transfer Enabled	Determines whether flash hook transfers can be performed on this line.	"Yes"	IA Line Configuration > Configuration>CO Supports Flash Hook transfer
Phone Number	The phone number associated with this line.	"555-1212"	IA Line Configuration > Configuration
Port Number	The analog board port number where this line is plugged in.	"1"	IA Line Configuration > Configuration
Prefix Digits	The digits required by the CO for calls made on this line.	"9"	IA Line Configuration > Configuration
Queue Identifier	Looking up this attribute will return the fully qualified queue identifier; however, this attribute does not actually exist in the Windows NT registry. The formula for this string is "Line Queue:" + the CIC line name.	"Line Queue:Line1"	(Internal)
Silence Auto Disconnect Enabled	Determines whether CIC or CO evaluates silence as an autodisconnect.	"Yes"	IA Line Configuration > Configuration
Silence Time	The number of milliseconds to wait before disconnecting when silence is detected.	"1000"	IA Line Configuration > Configuration
SMDI Enabled	Determines whether PBX has enabled SMDI support on this line.	"No"	IA Line Configuration > SMDI

Voice Resource Only			
snCreated	For system use only. Do not modify these values.		
snModified	For system use only. Do not modify these values.		

Line Group Key

Attribute Name	Description	Example	Set from...
Date Created	The date on which this line group key was created.	"908400570"	IA Line Group Configuration > History
Date Last Modified	The date on which an attribute in this line group was last modified..	"913150263"	IA Line Group Configuration > History
Lines	List of lines included in this line group.	"Line1" "Line2" "Line3"	IA Line Group Configuration > Configuration
Description	A textual description for the line group.	"This is a line group."	IA Line Group Configuration > Configuration
snCreated	For system use only. Do not modify these values.		
snModified	For system use only. Do not modify these values.		

DnisDid Key

Attribute Name	Description	Example	Set from...
Digits	The number the caller dials that associated with a certain queue identifier. (This number may not necessarily be the number dialed by the caller. The service provider may send a truncated number or a completely different number depending on the definition of service.)	"8794343"	System Configuration > DNIS/DID
Queue Identifier	The fully qualified queue identifier where the call is routed when the specified digits are dialed.	"Workgroup Queue:Marketing"	System Configuration > DNIS/DID
snCreated	For system use only. Do not modify these values.	_____	_____
snModified	For system use only. Do not modify these values.	_____	_____

Debugging Handlers

Debugging

Debugging allows you to monitor a handler as it runs and to stop it at locations or situations of your choosing. You can set a breakpoint on a particular step, for instance, and let your handler execute until it reaches that step. If you are interested in the values assigned to a variable, you can have the debugger pause execution ("break") whenever the handler changes the variable's value.

When you debug a handler, you are viewing the handler in read-only format. When the handler runs on the server, it pauses whenever it reaches a Notify Debugger step or any step that has been set as a breakpoint. Handlers that do not contain any Notify Debugger steps will pause at the initiator. You can then establish other breakpoints in the handler and let the handler execute until it reaches the next breakpoint.

In Debug mode, the [debug toolbar](#) and [debug palette](#) appear to allow you to manage the debugging session.

Interaction Designer's Debug mode is closely integrated with the Interaction Processor to support debugging over a network. This allows you to run Interaction Designer on a client workstation to debug a handler even though that handler is actually executing on the server.

Prerequisites for debugging:

- You must be running a copy of Interaction Designer connected to the server running the handler you want to debug.
- The handler or subroutine you want to debug must be [published](#) and running ([activated](#)) on the server.

Related Topics

[Debug a handler](#)

[Monitor the changes in the value of a variable in a debug handler](#)

[Setting a breakpoint](#)

[Stop the debugger](#)

[View the value of a variable in a debug handler](#)

Debug a handler

This procedure describes the process for debugging a handler.

1. Open the handler you want to debug.
2. Publish the handler (if not already published).
3. Choose Debug Handlers... from the Utilities menu
4. Choose the handler you want to debug and Click on OK.

Shortcut: you may also click on debug button to debug the published version of the currently displayed handler.

Designer will open a debug session window containing a read-only copy of the handler being debugged. This file has the extension .DBG. Steps that are breakpoints, i.e., Notify Debugger or the Initiator if the handler does not contain any Notify Debugger steps, plus any other steps that have been designated as breakpoints, appear in red. Also, the background of the design window will be wallpapered with the word "Debug" to further show that the active handler is running in debug mode.

When the handler is started on the CIC server and execution hits the first Notify Debugger or initiator breakpoint, the first breakpoint in the handler becomes highlighted and the handler execution pauses. You are now in debug mode. While in debug mode, you can see the values of variables in the Debug palette.

Note: When debugging a subroutine, if you set the breakpoint on the subroutine initiator, the values passed in by the calling handler are not set in the local handler variables until after the subroutine step executes. It's the execution of the subroutine step that assigns the variables.

Related Topics

[Debugging](#)

[Monitor the changes in the value of a variable in a debug handler](#)

[Setting a breakpoint](#)

[Stop the debugger](#)

[View the value of a variable in a debug handler](#)

Stop the debugger

While in Debug mode, you can stop the debugger to end the debug session for the current handler. If you stop the debug session, a prompt appears asking if you want to kill the handler. If you choose No, the handler continues to run to its normal completion. If you choose Yes, then all execution of the handler stops at that breaking point and does not continue to execute in IP. This behavior is configurable through the [Designer Preferences](#) page.

To stop the debugger:

To stop the debugger, you can do one of the following:

- Click the  button.
- Select Stop from the debug menu.
- Close the debug session window.

While execution is paused on a step, it is possible to change the execution point to another step. Right-click on the step from which you want execution to continue and select "Set Execution Point to This Step" from the pop-up menu. This allows users to step around known bugs during debugging.

Note: The execution point cannot be reset while the handler is paused on the initiator, nor can the execution point be set back to the initiator from elsewhere in the handler.

Related Topics

[Debugging](#)

[Debug a handler](#)

[Monitor the changes in the value of a variable in a debug handler](#)

[Setting a breakpoint](#)

[View the value of a variable in a debug handler](#)

Set a breakpoint

A breakpoint causes a handler running in debug mode to pause execution. While the handler is paused, you can check the values of variables. You can set breakpoints in two ways. First, when the debug session is initially started, you will be prompted to pick a Notify Debugger step or initiator to break on. After one of those initial breakpoints is hit, you can then set a breakpoint on other steps.

Your handler must be in debug mode for the following procedure.

To set a breakpoint:

1. Right-click on the step you want to set as a breakpoint.
2. Choose Set Breakpoint from the menu that appears. The step's color changes to red. When the debug handler runs, it will automatically stop at this step.

Similarly, you can remove a breakpoint by choosing Remove Breakpoint.

Related Topics

[Debugging](#)

[Debug a handler](#)

[Monitor the changes in the value of a variable in a debug handler](#)

[Stop the debugger](#)

[View the value of a variable in a debug handler](#)

Viewing the Value of a Variable While Debugging

By default, the [Debug palette](#) appears at the bottom of the window at the start of a debugging session. The first tab on this palette is the Watch Variables tab. This window displays the values of watch variables throughout the debugging session and also gives you the option to have the handler pause whenever the value of one of the watched variables changes.

When the value of a watched variable changes, the background color changes for that variable in the watch window.

When the debugging session is first started, the watch window is empty. There are several ways to add variables to it:

- You can drag variables into the Watch window directly from the Variables palette. Simply select the variable or variables you want to monitor from the Variables palette and drag them into the Watch palette.
- You can also add variables to the Watch window from the Variables palette by right-clicking on them and selecting "Add Selected Variable(s) to Watch Window."
- You can add variables by clicking the "< Click to Add >" entry in the listview. When this is done, a combo box appears from which users can select the variables they want to watch.
- Users can right-click on the watch window itself to display a pop-up menu that allows them to add variable watches. Select the Add menu item to display the drop-down combo box from which users can select the variables they want to add.

You can add, change, or remove watch variables from the watch window by right-clicking them and selecting the appropriate option from the pop-up menu.

If the watch variable value is too large to fit in the Value column, users can click on values displayed in the Value column to display the value in the Variable Value Details View dialog.

When viewing the value of a native type variable (i.e., String, Integer, Boolean, DateTime, Numeric), users can dynamically update the value of the variable while debugging on this Details Window. By default, the dialog creates an expression for the variable when the dialog is displayed. If the user wants to change the value of the variable, they can do so in the expression editor control and then click on the Set Value button.

Note: setting the value of a variable is only supported for native type variables in a debug session.

When execution is paused on the initiator in a debug session, handler variable values will have their initial values set. This is because it is the execution of the initiator step that assigns the initiator output values to the handler variables. All parameters passed in to a subroutine (either input only or input/output) are assigned *after* the subroutine initiator runs.

Related Topics

[Call Stack](#)

[Debug a handler](#)

[Debug Palette](#)

[Debugging](#)

[Handler Variables](#)

[Monitor the changes in the value of a variable in a debug handler](#)

[Setting a breakpoint](#)

[Step Variables](#)

[Stop the debugger](#)

[Watch Variables](#)

Monitor changes in the value of a variable in a debug handler

While in debug mode, you can watch for a change to the value of one or more variables. Interaction Designer checks for changed variables at each breakpoint, and will highlight any watched variables that have changed since the previous breakpoint in the [Watch Variables](#) window in the [Debug Palette](#).

If you want the handler to pause when a watch variable changes, select the "Break on Change" box for that variable in the watch window. If the variable's value changes, the handler will pause execution at the next step. Note that because the variables don't change until a step exits, the step that execution pauses on is not the step that changed the variable. The variable was changed by the previous step.

Related Topics

[Debug a handler](#)

[Debugging](#)

[Setting a breakpoint](#)

[Stop the debugger](#)

[View the value of a variable in a debug handler](#)

Working with Subroutines

Introduction to Subroutines

Subroutines are handlers started by other handlers, as opposed to handlers started by events on the CIC server. Subroutines are useful for many reasons.

First, a subroutine nicely encapsulates a set of logic into a single callable action. Thus, you can "reuse" a subroutine by calling it from any handler where that set of logic is appropriate. For instance, suppose that you have created a subroutine called `LookupAccountNumber`. Given a person's name, that subroutine will open a customer database, look up that person's account number and return the account number to the handler that called it. Now, imagine a system of handlers that deals with customer support. In such a system, the need to retrieve customer account numbers is a common one. Thus, when you build the customer support system, you would be able to call `LookupAccountNumber` from any handler that needs such information.

Second, subroutines ease your maintenance burden. Recall the first example of `LookupAccountNumber`. Suppose that, after your customer support handlers are finished and working, the customer database is redesigned. Instead of directly tying customer name to account number, the database has been changed to tie customer name to social security number, and then, in a separate table, to tie social security number to account number. Since your `LookupAccountNumber` localizes the access to the database in a single handler, you only need to change how `LookupAccountNumber` gets the information. Essentially, you'll change it to use the given name, retrieve a social security number, and then use the social security number to retrieve the account number. This change can be made without any changes to all the other handlers that call `LookupAccountNumber`.

Third, subroutines make handler building more visually manageable and understandable. By localizing groups of steps into subroutines, you can cut down on the number of steps that appear in the calling handler. Handlers with hundreds of steps can be very hard to understand and manage; a few subroutines can make a large handler easy to modify and understand. Essentially, by using subroutines properly, you break the logic down into manageable pieces.

How do I create a subroutine?

Creating subroutines is almost exactly like creating handlers. There are two differences.

The first difference is that every subroutine starts with a [subroutine initiator](#). In this subroutine initiator, you'll define what information is passed into the subroutine from a handler that calls it. You'll also decide what information can be passed back to the handler that calls the subroutine. These pieces of information exchanged between a subroutine and its caller are called parameters. See [Add or edit a subroutine parameter](#) for more information.

The second difference appears during the [publish process](#). If you are publishing a subroutine, you need to select or enter a subroutine category on the Publish Handler dialog. During the publish process, Interaction Designer creates a subroutine tool for the [design palette](#).

How do I call a subroutine?

When a subroutine is published, a tool for that subroutine appears on the subroutine palette. To create a call to the subroutine from any handler, simply drag the subroutine tool from the subroutine palette and drop it onto the handler.

Loading and Viewing Subroutines

Subroutines may be saved and loaded into Interaction Designer in the same manner as other handlers. To ease handler authoring, however, there are two additional ways in which subroutines may be loaded and viewed:

1. When viewing a handler containing a subroutine step, you may right-click on that step and select "Download from Server" from the pop-up menu. This will download the subroutine to the directory specified in the [Designer Preferences](#). If no directory is specified there, the user will be prompted for one. Once the subroutine is downloaded, it will open in a new window.
2. From the Subroutine page of the Design Palette, you can right-click on a subroutine icon and select "Download from Server" from the pop-up menu. This will also download the subroutine to a specified directory and open it in a new window.

The Published version of the subroutine that is running on the server will always be the version that is downloaded.

Related Topics

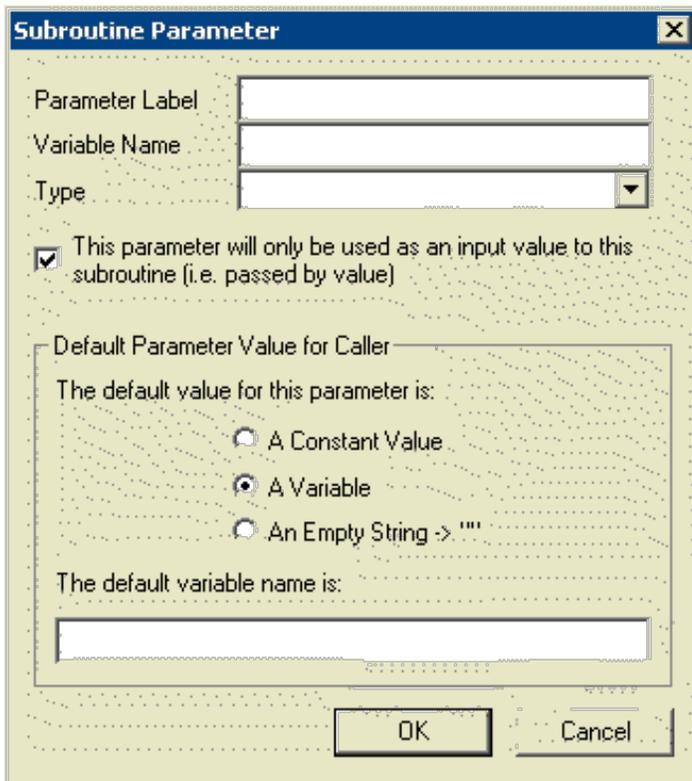
[Add or edit a subroutine parameter](#)

Add or edit a subroutine parameter

Subroutine parameters can be added and deleted from the Properties notebook of a [Subroutine initiator](#). Add or edit a subroutine parameter any time you need to capture information from the handler that calls the subroutine. The parameters that you enter using this procedure appear in the subroutine tool that appears on the subroutine palette. The parameters do not appear in the step that calls the subroutine until you publish the subroutine.

To add a subroutine parameter:

1. Open the Subroutine Initiator to the Parameters page.
2. Click the Add button. The Subroutine Parameter dialog appears, as shown in the following figure.



3. Enter a parameter label. This label appears in the step that calls this subroutine.
4. Enter a variable name to hold the value entered for the parameter.
5. Choose a type for the variable.
6. Select or clear the **This parameter will be used only as an input value to this subroutine** check box. If you want the parameter to pass a value back to the handler that called it, make sure the check box is not selected. Selecting this check box means that the value of this parameter cannot be changed within this called subroutine. Deselecting this check box means that the value of this parameter can be changed within this subroutine and changes will be passed back to the handler that called this subroutine.
7. If you want a default value for the parameter, type it in the Default entry dialog box. This can be a constant value, a value of a variable, or an empty string.
8. Click OK to add this subroutine parameter.

To edit a subroutine parameter:

1. Open the Subroutine Initiator to the Parameters page.
2. Click the Edit button. The Subroutine Parameter dialog box appears.
3. Make any necessary changes to the settings.
4. Click OK when you are finished.

Managing Handlers

Manage Handlers notebook

In the Manage Handlers notebook you can activate and deactivate Primary and Monitor handlers. All published handlers are listed here. Once you publish a handler, it appears in the Inactive Handlers list, unless it is already an actively running handler. CIC begins to use Activated handlers as soon as another event occurs for which the handler is registered. If Interaction Processor is currently running older versions of the handler, those threads finish before the new handler is used. Any threads running on handlers you deactivate continue until the thread is finished.

Caution: CIC cannot differentiate between primary and monitor handlers in the Manage Handlers Notebook. This is why the lists of inactive handlers are identical on the Primary Handlers and Monitor Handlers page. Make sure you don't activate a primary handler on the Monitor Handlers page, or vice versa.

Note: You can also manage handlers in Interaction Administrator's Server Configuration container.

Primary and Monitor Handlers

Primary handlers are generally handlers that act directly on objects such as calls within the system. Only one primary handler can be activated for a given initiator. For example, you cannot activate two primary handlers that both start with the Incoming Call initiator. Only one handler can act on the incoming call. This prevents two handlers from performing disparate actions on a single object. If you attempt to activate two handlers that start with the same initiator, CIC generates an error message in the event log.

Monitor handlers do not actively manipulate or modify objects in the system. Typically they retrieve call attributes and write that information to a database for reporting purposes, although they are not limited to reporting. CustomCallDisconnectMonitor determines if a call was recorded, and if so, where to send a copy of that recording. Since monitor handlers are not acting on objects, more than one monitor handler can use the same initiator. For example, CallDisconnectMonitor and CustomCallDisconnectMonitor both use the Call Monitor Initiator configured to start when a call disconnects.

Activating and Deactivating Handlers

Click on one of the links below for more information on activating primary and monitor handlers.

[Primary Handlers page](#)

[Monitor Handlers page](#)

Monitor Handlers page

Monitor handlers do not actively manipulate or modify objects in the system. Typically they retrieve call attributes and write that information to a database for reporting purposes, although they are not limited to reporting. CustomCallDisconnectMonitor determines if a call was recorded, and if so, where to send a copy of that recording. Since monitor handlers are not acting on objects, more than one monitor handler can use the same initiator. For example, CallDisconnectMonitor and CustomCallDisconnectMonitor both use the Call Monitor Initiator configured to start when a call disconnects.

Caution: CIC cannot differentiate between primary and monitor handlers in the Manage Handlers Notebook. This is why the lists of inactive handlers are identical on the Primary Handlers and Monitor Handlers page. Make sure you don't activate a primary handler on the Monitor Handlers page, or vice versa.

Note: You must have manage handler rights (configured in Interaction Administrator) to activate and deactivate handlers.

Inactive Handlers

Handlers in this list are registered on the server but not active.

To activate a monitor handler:

- Double-click on the inactive monitor handler's name.
- or
- Select the inactive monitor handler and click the Add button.

Active Handlers

Handlers in this list are registered with the Interaction Processor and currently active (monitoring events in the Interaction Processor).

To deactivate a monitor handler:

- Double-click the active monitor handler name.

or

- Select the active monitor handler and click Remove.

Primary Handlers page

Primary handlers are generally handlers that act directly on objects such as calls within the system. Only one primary handler can be activated for a given initiator. For example, you cannot activate two Primary handlers that both start with the Incoming Call initiator. Only one handler can act on the incoming call. This prevents two handlers from performing disparate actions on a single object. If you attempt to activate two handlers that start with the same initiator, CIC generates an error message in the event log.

Caution: CIC cannot differentiate between primary and monitor handlers in the Manage Handlers Notebook. This is why the lists of inactive handlers are identical on the Primary Handlers and Monitor Handlers page. Make sure you don't activate a primary handler on the Monitor Handlers page, or vice versa.

Note: You must have manage handler rights (configured in Interaction Administrator) to activate and deactivate handlers.

Inactive Handlers

Handlers in this list are registered on the server are but not active.

To activate a primary handler:

- Double-click on the inactive primary handler's name.

or

- Select the inactive primary handler's name and click Add.

Active Handlers

Handlers in this list are registered with the Interaction Processor and currently active (available for use by the Interaction Processor).

To deactivate a primary handler:

- Double-click the active primary handler name.

or

- Select the active primary handler and click Remove.

Handler Best Practices

Handler Best Practices

The following links lead to topics that contain information, including tips and tricks, that should be useful to you in managing your own handlers:

[ACD Queue Tips](#)

[Database Practices](#)

[Documentation Practices](#)

[Efficiency Practices](#)

[General Handler Authoring Practices](#)

[Limiting the Impact on Interaction Processor](#)

[Overall Project Tips](#)

[Overview of Building Handlers and Subroutines](#)

[Prompts](#)

[Troubleshooting Tips](#)

General Handler Authoring Practices

At the beginning of every handler

- Insert a Write Trace step that identifies the handler name and that it started. The IP.log trace log only shows when a handler finishes. It does not indicate when a handler starts.
- Assign blank or null values to all assignment steps.

Build to the right

Design your handler so that the steps flow down and fill the screen but not so far that you have to scroll down to see the steps. Then connect and start over at the top. As the handler gets larger, add columns of steps to the right. This makes it easier to scroll through the handler since you only have to scroll to the right and not down. A variation would be to reverse the layout and make it so you scroll down but never to the right to view the handler.

Double monitors and/or high resolutions

Get a video card that can support dual monitors or high-resolution setting to minimize the amount of scrolling you'll need to do.

Extend step to show whole selection step

Expand step size to show the values and the step label.

Use one Selection step instead of multiple condition steps

This may seem obvious but it is a common mistake.

Identify data type of variables in name

Use "Hungarian Notation" to prefix variable names with the type of variable. For example, a string variable name would be prefixed with "str". A list of string would be "Istr". Look at the default handlers for examples. The Queue Period Stats handlers are a good example.

Link steps as integers take up less space

Many handler authors use "link" steps to clean up link lines. The link steps are usually just dummy assignment steps for a variable called Link. Instead of creating Link as a string variable, make Link an integer. Integers take up less space.

Store information subject to change outside of handlers

Design handlers so as to avoid making it necessary to open the handler to make small changes. Use server parameters, IA tables or database tables to store the information like directory paths, file names, server names, external numbers for blind transfers, or anything else that may need to be changed.

Copy & paste to new handler to get rid of unused variables

After making many changes to a handler, select all the steps (Ctrl+A) and then copy and paste the steps into a new handler. This will remove any un-used variables and will renumber the steps.

No need to reboot server to initialize accumulators

Just end the accumulator server using the Task Manager and let it restart on its own.

Create custom attributes to store default attributes that are not stored

Certain default call attributes such as ANI/DNIS string are not populated by default. Make a point of initializing these call attributes.

Related Topics

[ACD Queue Tips](#)

[Database Practices](#)

[Documentation Practices](#)

[Efficiency Practices](#)

[Handler Best Practices](#)

[Limiting the Impact on Interaction Processor](#)

[Overall Project Tips](#)

[Overview of Building Handlers and Subroutines](#)

[Prompts](#)

[Troubleshooting Tips](#)

Documentation Practices

Use File/Properties to document handler

Instead of using notes for the initiator, use the File Properties dialog to document a handler.

Use assignment steps to document a series of steps

Use an assignment step labeled "ReadMe for steps X to Y" to document a series of steps.

Paste statement into step notes

Paste the statement value into the step notes in order to document the condition and to make editing easier. Make changes to the statement in the notes and then paste the changes into the statement value.

Indicate changes in step label

Anytime you make changes to a production handler or a handler that someone else wrote, indicate which steps you changed by changing the step label. Add an "*" before the label along with your initials. Also indicate the changes you made in the step notes. You or someone else can then easily find any changes made to the handler by using the [QuickJump](#) utility. Steps with "*" labels will be at the top of the list.

Related Topics

[ACD Queue Tips](#)

[Database Practices](#)

[Efficiency Practices](#)

[General Handler Authoring Practices](#)

[Handler Best Practices](#)

[Limiting the Impact on Interaction Processor](#)

[Overall Project Tips](#)

[Overview of Building Handlers and Subroutines](#)

[Prompts](#)

[Troubleshooting Tips](#)

Efficiency Practices

Minimize lookup steps by writing values to call attributes

Avoid repeatedly looking up Directory Services/Registry information by assigning the information to call attributes. Simply retrieve the attribute at any time in any other handler when you need the information again about the call.

Use accumulators as fast lookup variables

If the database state does not change often, make your database driven handlers more efficient by avoiding lookups for every call by storing the state in accumulators and then looking at the value of the accumulator for every call. A good example of this would be storing queue open and closed times in a database and then creating a handler that periodically does the database lookup and stores the value in an accumulator. These handlers would deal with incoming calls then only look at the accumulator rather than performing the disk intensive DB lookup themselves.

This type of design is especially useful for systems that handle large volumes of calls. A variation of this design is to use registry entries to store and retrieve the values. The point is to try to minimize disk intensive tasks with every call.

Related Topics

[ACD Queue Tips](#)

[Database Practices](#)

[Documentation Practices](#)

[General Handler Authoring Practices](#)

[Handler Best Practices](#)

[Limiting the Impact on Interaction Processor](#)

[Overall Project Tips](#)

[Overview of Building Handlers and Subroutines](#)

[Prompts](#)

[Troubleshooting Tips](#)

Overall Project Tips

Protecting default handlers and customized handlers

- Keep custom handlers in their own separate directory.
- Write-protect the default system handlers either at the file or directory level to avoid inadvertent changes.
- Store modified, default system handlers separately from the default handlers and any custom handlers.
- **Avoid making any changes to the default handlers.**
- If you must make changes, use subroutines whenever possible.
- If you find that you always end up making changes to certain default system handlers, request the addition of a customization point by Genesys.

Backup customer handlers in office

If you are creating handlers for a third party, don't count on your customer to be able to adequately back up custom handlers themselves. Maintain copies of their handlers off site. This also protects you from any changes they might make to the handlers you have written for them.

Handler naming using company name prefix

When creating handlers for a third party, name customer-specific handlers with a common prefix in order to more easily sort, publish and manage them. However, you should avoid attaching specific company or project names to handlers in case you need to reuse the handler for another project.

Reuse code across projects

Avoid naming handlers and variables and other items after clients or projects so that you can easily reuse the handlers for other clients or projects. This is especially true with subroutines that act like functions.

Handler Design Project Management Tips

Plan out handlers using call flow Visio diagrams. Make sure all paths for all options in the call flow end with specific requirements. A common mistake is to leave most of the detail out of the minor options. Get a sign-off on the call flow scope of work before starting. No work should be done without a signed call flow.

Use server parameters for information specific to site

Use server parameters to customize handlers for certain projects or provide specific names to handler items.

Related Topics

[ACD Queue Tips](#)

[Database Practices](#)

[Documentation Practices](#)

[Efficiency Practices](#)

[General Handler Authoring Practices](#)

[Handler Best Practices](#)

[Limiting the Impact on Interaction Processor](#)

[Overview of Building Handlers and Subroutines](#)

[Prompts](#)

[Troubleshooting Tips](#)

Database Practices

Use views and stored procedures established for date time conditions

Stored SQL procedures and views execute much faster than queries especially in the case of date time conditions

Use the Logging Custom Passthrough tool for all database writes

Using the regular DB tools to write data to tables can fail if the connection to the database is down. By using the [Logging Custom Passthrough](#) tool you can take advantage of the built in support for MSMQ in CIC and assure that your writes will be processed when the connection is restored. The only limitation is that you must create your own table in the CIC database. You can't use the LCP tool to write to other databases.

Are all database connections closed when a handler is stopped? Should the connection be released/closed?

All handlers should contain cleanup code that will be called when the handler exits. Every handler that creates a DB connection will register a cleanup code to release it if it is not explicitly released in the handler.

In earlier releases of CIC, it was possible to cause the reference count on the DB to get improperly set when the connection handle was passed to a subroutine. This caused it to not be automatically cleaned up if the handler were to exit unexpectedly. Even though this problem no longer exists, we still feel people should explicitly release the connection themselves in the handlers, just to be extra safe and to promote good programming habits.

Is the number of database connections unlimited ?

In theory, yes. The default of 64 database connections is only a default. This can be increased by setting a command-line parameter to IPDBServer (when running as a service, you set this in the registry under the CIC process tree entry for IPDBServer). However, 64 connections is a huge number of connections. If you are finding that you need more, you probably have some long-running transactions somewhere that need to be optimized. Although you can raise the max connection limit, keep in mind that in reality there really isn't such a thing as a truly 'unlimited' number of connections. DB connections take a lot of resources - both on the client and on the server. So you are going to hit a practical limit at some point. What exactly that limit is depends on the machines involved and how much degradation you are willing to live with, but somewhere around 100 connections would be a good point to start being concerned.

Is it true that DB's must be opened and closed within a handler so connections are not passed through as subroutine parameters? Instead, should new connections be made in every handler?

Again, this isn't a requirement - we do support passing connection handles to subroutines. However, doing so usually violates the principle of not holding onto valuable resources (in this case, DB connections) any longer than you absolutely must. If the operations performed in the subroutine are logically part of the same transaction as the caller's, then you should question why they are encapsulated in the subroutine. If they are an independent transaction, then they should get and release their own connections. Remember, with connection caching getting a connection (after the first initial connection) is a very fast operation. Connection caching does change the way you do things, but it is worth it. Besides the obvious benefit of virtually eliminating the overhead of constant connecting/disconnecting, it also promotes good transactional programming style. In a very simplified format, here is one effective approach within CIC:

1. Identify all of the individual database transactions needed. A transaction for this discussion means a series of database operations that must either all be completed (committed) successfully, or all be completely undone (rolled-back). In other words, the series of operations must appear to be one single operation that either succeeds or fails (i.e. no partial succeeding). Implicit in this definition is the requirement that the operations in the transaction must be intricately linked together in order to preserve database consistency. In other words, if they are not intricately linked together, then they should be separated into their own transactions.

Note that a transaction of just one operation is legitimate; in fact, this is the most common type of transaction. From a theoretical standpoint, whether a transaction is composed of one operation or several is irrelevant. However, from a programming standpoint it is. Most RDBMS guarantee that an individual operation, like an INSERT statement, will behave atomically (the behavior when triggers are introduced does vary, though). But if you want this atomic behavior for [multiple](#) operations, you must inform the RDBMS of this. In CIC, you would issue a BEGIN TRANSACTION command (via [DB SQL Exec](#)), followed by either a COMMIT TRANSACTION or ROLLBACK TRANSACTION when you are done.

People will sometimes group unrelated or semi-related operations together under one transaction when they really belong in separate transactions. This is often done out of convenience, or out of an attempt at optimization. Try hard not to do this. When you are identifying transactions, the question you need to ask yourself is "will my database **really** be in an inconsistent state if I split these operations apart and one of them succeeds and one fails?" If the answer to that question is no, then split them into separate transactions.

2. Every transaction - whether composed of a single operation or multiple operations - should get & release its own connection. The amount of time the connection is held should be minimized as much as possible. I.e., get it just prior to using it, and

release it *immediately* afterwards.

3. Implement each transaction in its own separate subroutine. The subroutine should have no other purpose other than to execute the database transaction, returning any result data back to the caller. It should not be doing any other significant processing. While this may at first seem hard to accomplish, cases where you legitimately can't do this are exceedingly rare.

A common situation where people don't think they can do this is when they need to fetch multiple rows of data and perform some work on each row. However, in almost all cases, fetching everything into a list first will solve this.

One advantage to putting all your database code in separate, usually small, dedicated subroutines is that if your database schema changes you will only need to make any corresponding handler changes in a small set of well-isolated subroutines. Even better, if you decide to implement some transactions in stored procedures, the changes would only involve those subroutines. Also, one of the biggest benefits of using dedicated subroutines is that it helps enforce the habit of identifying and encapsulating independent transactions.

Does all this opening and closing take additional resources?

No, since we are not actually opening & closing connections, but instead getting & releasing connections to/from the cache.

Related Topics

[ACD Queue Tips](#)

[Documentation Practices](#)

[Efficiency Practices](#)

[General Handler Authoring Practices](#)

[Handler Best Practices](#)

[Limiting the Impact on Interaction Processor](#)

[Overall Project Tips](#)

[Overview of Building Handlers and Subroutines](#)

[Prompts](#)

[Troubleshooting Tips](#)

Troubleshooting Tips

Use Log Event tool after Failure path

Add [Log Event](#) steps after most failure paths in order to log the error to the NT Event log. Errors logged this way are much easier to find than looking at trace logs. Use the "Error" log type sparingly. Use the "Warning" type instead unless the error is really bad. Identify the handler name in the text of every Log Event step by inserting a variable called HandlerName in the text. Also identify the previous step number that triggered the log event and a brief description of the possible problem. Please note that some telephony tools will follow the failure path even when no "real" failure occurred. The [Extended Get Key](#), for instance, will take the failure path if the caller hangs up or is disconnected while the [Get Key](#) is playing prompts or waiting for input.

Use "Informational" log events to troubleshoot or verify handler functionality

"Informational" NT Event log messages can be created in a handler using the Log Event tool. Use them to verify important steps in a handler or track major changes or unique events in a handler. Can also be used to troubleshoot handlers by identifying failure paths or other unwanted behaviors. Use this method rather than debugging handlers or looking at trace logs. Create [Assignment](#) steps with HandlerName to find errors in Event logs more easily.

Use blank assignment steps at the end

For handler paths that end with multiple possible conditions or selections, insert a blank assignment step in order to facilitate troubleshooting. When debugging without these steps, it can be hard to determine which path is taken.

Set breakpoint in debug to skip over sections

You can turn any step into a breakpoint after a handler is already in debug mode. Your handler must already be in debug mode to set a breakpoint:

1. Right-click on the step you want to set as a breakpoint.
2. Choose Set Breakpoint from the menu that appears. The step's color changes to red. When the debug handler runs, it will automatically stop at this step.

Use a condition step for cell phone ANI before debug step

To debug a handler on a live system that is receiving many calls, insert a [Condition](#) step before a [Notify Debugger](#) step that checks the ANI call information and matches it to your cell phone number. This will allow you to debug a live call without risking interrupting a customer call. Only your cell phone number will trigger the debug. Consider using a system parameter for the ANI string to avoid having to republish the handler when you want to change the debug trigger number.

Debug using custom notification

Trigger your subroutine handler for debugging using a [Send Custom Notification](#) command. Create a separate handler that has the [Custom Notification](#) initiator followed by your subroutine.

Use a Net Send to troubleshoot handlers

Insert an [Execute Shell Command](#) step with a Net Send command to quickly be notified of a troubleshooting event in a handler. Another option is to use [email tools](#) and send the event to yourself.

Learn how to read IP trace logs

Every handler developer should be familiar with how to read the information in the IP trace logs. This can often be the only way to troubleshoot handler problems. Often a handler will work when debugged, but fail when running in normal mode. An IP log will tell you exactly what was happening when a handler runs.

Write specific troubleshooting to trace logs

Use the [Write Trace Message](#) tool to indicate specific events such as conditions and selections you want to track in a problem handler. These steps are not shown in the trace log even at the highest "Notes" level of tracing.

Use a special viewer to read trace logs

Trace log files can often be quite large and take forever to open using regular word processors. PureConnect Customer Care uses Visual Slick Edit. It can quickly open large files and can do advanced searches.

Related Topics

[ACD Queue Tips](#)

[Database Practices](#)

[Documentation Practices](#)

[Efficiency Practices](#)

[General Handler Authoring Practices](#)

[Handler Best Practices](#)

[Limiting the Impact on Interaction Processor](#)

[Overall Project Tips](#)

[Overview of Building Handlers and Subroutines](#)

[Prompts](#)

Prompts

Prompt Scripts

Have the customer fill out a form for exactly what they want. This form should include not only each specific prompt that the customer wants, but also how the customer wants each prompt to be worded.

Record Prompts using phone with a high quality headset

When it is not possible to use a professional recording studio, prompt recordings should be done over the phone using a high quality, noise-canceling headset. Record the prompts as a voicemail to yourself and save the voicemail attachments.

Note: The format of the resulting voicemail message depends on the voicemail compression settings in Interaction Administrator. Depending on the settings, you might need to convert the files to the appropriate format. CIC IVR prompts require the format CCITT, u-Law, 8 kHz, 80-bit, mono. Recordings made in Interaction Attendant are in the u-Law format.

Related Topics

[ACD Queue Tips](#)

[Database Practices](#)

[Documentation Practices](#)

[Efficiency Practices](#)

[General Handler Authoring Practices](#)

[Handler Best Practices](#)

[Overall Project Tips](#)

[Overview of Building Handlers and Subroutines](#)

[Troubleshooting Tips](#)

ACD Queue Tips

Transfer ACD calls to a separate queue for voicemail, external transfers or other special processing

ACD calls that need to go to voicemail or that require any other type of special additional IVR processing should be transferred to special dummy queues. This will clean up the statistics for the parent queue and make reporting easier.

For example, calls that time out while on hold in an ACD queue that then should be transferred to a cell phone should first be transferred to a special handling queue. In this case, create a queue called XFER and transfer the call there first. Then use [Blind Transfer](#) or a similar step to transfer the call to an external number. This will keep the time spent on those calls from affecting the stats of the parent queue.

Related Topics

[Database Practices](#)

[Documentation Practices](#)

[Efficiency Practices](#)

[General Handler Authoring Practices](#)

[Handler Best Practices](#)

[Limiting the Impact on Interaction Processor](#)

[Overall Project Tips](#)

[Overview of Building Handlers and Subroutines](#)

[Prompts](#)

[Troubleshooting Tips](#)

Limiting the Impact on Interaction Processor

When customizing or authoring handlers, be aware that each custom event or initiator causes overhead for Interaction Processor's thread count. While new Call or Interaction Disconnect, TCP/IP, SOAP, or CustomNotification handlers might work fine in testing or under a medium load, the thread count under a high load could be substantially higher than what CIC has been tested with in QA labs.

Here are some suggestions for limiting the impact on Interaction Processor:

- Use existing customization points for Interaction or Call disconnect when possible.
- When designing handlers, consider the maximum number of threads in the thread pool, which is set in Interaction Administrator. For more information, see the topic "IP Configuration" in the Interaction Administrator online help. The limit is set in Interaction Administrator after the handler publishes. If the number of events received exceeds the set limit, the messages are queued and run in order as the custom handlers terminate. Design for the possibility of queuing and failures due to timeouts to minimize the impact of activity surges on call processing availability.
- Do not excessively retry events that initiate handlers. If an event timed out, it could be because it was queued. A retry would result in two of the same event in the queue and would slow down the processing.
- Limit the runtime of custom handlers. Require that handlers run quickly or fail due to timeout.
- If you can perform an action from a CIC client or from Scriptor, do so. When you send actions to Interaction Processor, you might gain access to convenient tools or database integrations, but it puts a scaling bottleneck into the design that is easily avoided if each individual client performs their own action.
- Avoid excessive round trips of data with TCP/IP tools. These tools are good ways to get custom data to clients, but if that involves data that must be updated on the server with TCP/IP data back to the CIC server, be careful not to do that too frequently. An alternative is to use web services to update the data on a third server. You can have the CIC server send the initial data there along with its updates, and then have the clients send their data there along with updates to avoid the load on the CIC server.

Related Topics

[ACD Queue Tips](#)

[Database Practices](#)

[Documentation Practices](#)

[Efficiency Practices](#)

[General Handler Authoring Practices](#)

[Handler Best Practices](#)

[Overall Project Tips](#)

[Overview of Building Handlers and Subroutines](#)

[Prompts](#)

[Troubleshooting Tips](#)

Working with Steps

Introduction to steps

Each step in a handler or subroutine represents an action that a handler performs when run. Steps are like functions in a program. A step is a single instance of a tool dragged from the tool palette into the handler window. Steps appear as named boxes in the handler and are connected to other steps by links. Steps represent a single action that takes place when a handler or subroutine runs. For example, an Assignment step can assign a value to a variable. A Whitepages step can look up a telephone number in the CIC Whitepages. A Condition step can evaluate a condition and decide what step should come next. A Log Message step can write information to a log. Each step performs an action in the handler, and combinations of linked steps make up the handlers that respond to an event that occurred on the CIC server.

Anatomy of a step

The items in the following figure are described in the paragraphs below.



- **Step type**
The type of tool from which the step was created. For example, if the step was created from the Assignment tool, the step type is Assignment.
- **Step label**
The custom name a handler author typed for this tool. Use these labels as quick reminders of a step's purpose in a handler. If you do not enter a name for a step, the step label is the same name as the step type. For example, if you do not type a unique label for a Log Message step, Log Message appears in the Label for the step.
- **Icon**
The unique icon that corresponds with the step's type. For example, all Generate HTML steps have the same icon as the Generate HTML tool. You cannot change a step's icon.
- **Exit Paths**
The paths a step can take as a result of a step's action. It is through these exits that steps link to other steps. For example, a Condition step has two exit paths: True and False. If the condition evaluated by the Condition step is true, the step will exit along the true path. If the condition evaluated by the Condition step is false, the step will exit along the false path.

If a step has no exit paths, or no links from an exit path, the handler or subroutine ends after the completion of that step.

- **Properties notebook**
The notebook from which you can define the action a step performs. Each step has a General page where you type the step's label and any special notes about the step. Each step also has at least one of the following pages: a Settings page, an Inputs page, and an Outputs page. It is on these pages that you customize the properties of the action that step will perform.

For example, the Assignment step assigns a value to a variable. From the Properties notebook for that step, you specify the variable and the value it is assigned.

The Inputs page of a step's Properties Notebook lists all the parameters needed by the step to perform the action. Some of these parameters may be optional. The Outputs page of a step's Properties Notebook displays the values that are created as a result of the step performing some action. If these output values are stored in variables, they can be accessed by other steps in a handler.

For a description of the properties and pages of each step, see the [Tools](#) section in the reference section of this online documentation. You can also click the help button inside a step's Properties Notebook for help on that tool.

Add a step

Use one of the following procedures to add a step to a handler.

Drag and Drop Method:

1. Click on a tool on the Tools page of the Design palette.
2. Drag the tool into the handler window.

Double-Click Method:

Double-click on a tool on the Tools page of the Design palette.

The new step for that tool appears in the top left corner of the handler window. If you do not see the step, it may be covered by a step already in the top left corner. Move the covering step to reveal the step you added.

Delete a step

You can delete a step or a group of steps. Select a group of steps by drawing a box around them with the mouse pointer. The selected steps are individually highlighted.

Note: There is currently no way to undelete a step or group of steps.

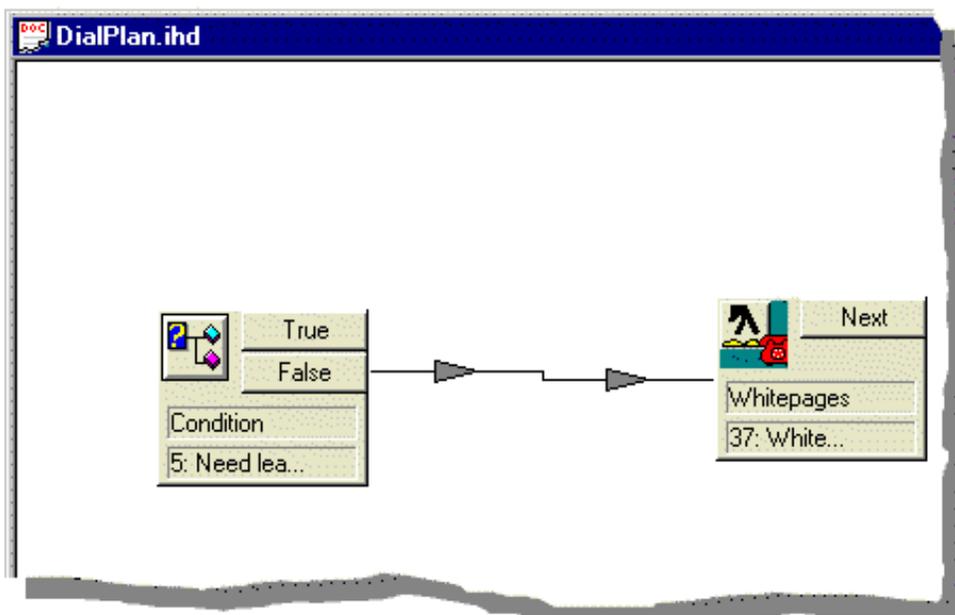
To delete a step or group of steps:

1. Click on the step (or group of selected steps) you want to delete.
2. Press the delete key on your keyboard.
The steps are deleted from the handler window.

Connect a step

A step may have up to one link for each exit path. An exit path without a link causes the handler or subroutine to end if that exit path is taken. Connected steps stay connected, even if they are moved. If you **copy** two or more connected steps, the copies are also connected when you paste the steps. The figure below shows two linked steps.

1. Click the exit you want to connect from.
2. Drag to the step you want to connect.
A link appears between the exit path and the next step.



Move a step

You can move a step (or group of steps) to any location in a handler window without breaking its links or affecting its function. Select a group of steps by drawing a box around them with the mouse pointer. The selected steps are individually highlighted. It's a good idea to locate the steps in the order they occur to make the handler's actions easier to understand.

To move a step:

1. Select the step or steps.
2. Drag the step or steps to the new position.
The handler window scrolls automatically if you drag the steps beyond the handler window's edge.

Edit a step

You can edit the properties of a step by opening its Properties notebook. The Properties notebook for a step contains one or more parameters. Edit these parameters to customize the action a step will take when the handler is run. Changes you make to the properties of a step will not be used by CIC until you [publish](#) the handler or subroutine containing the step.

1. Double-click the step.
or
Right-click on the step and choose Properties from the menu that appears over the step.

The Properties notebook appears. While the Properties notebook is open, you cannot perform other actions in Interaction Designer.

2. On the General Page, you can change the Label and the Notes.
3. On the other pages, you can change the values of any parameters.
4. When you are finished, click OK.
If the Incomplete Required Parameter dialog box appears, you have left a parameter empty and should click Yes to return to the Properties notebook to fill in the value.

View the properties of a step

Use the following procedure to view the properties of a step.

- Double-click the step.
or
- Right-click on the step and choose Properties from the menu that appears over the step.

The Properties notebook appears. While the Properties notebook is open, you cannot perform other actions in Interaction Designer. If you want view the properties of another step, you must first close this Properties notebook.

Related Topics

[Edit a step](#)

Find a step

Some handlers and subroutines are too large to fit entirely in the editing window. The [QuickJump](#) function can save scrolling time by allowing you to jump directly to a step in a handler or subroutine.

To find a step:

1. From the View menu, choose QuickJump. The QuickJump palette appears. All of the steps in the handler are listed in the Step Label list.
2. Double-click on a step listed in the QuickJump palette to move focus to that tool in the design window. The QuickJump dialog box disappears, and the selected step appears in the center of your editing window.

To find steps of a specific type

There are two ways to find all occurrences of a specific type of step (for example, all Condition steps) in a handler.

1. right-click on a step of that type inside the handler and select "Show Steps of This Type..." from the popup menu; or
2. right-click on the icon for the desired step in the tool palette and select "Show Steps of This Type..." from the popup menu.

Either way, the QuickJump palette will appear, listing all occurrences of that step in the handler.

Copy a step

You can copy a step or group of steps, and then paste those copies into the same handler or another handler. Select a group of steps by drawing a box around them with the mouse pointer. The selected steps are individually highlighted. The steps you paste are linked in the same manner as the steps you copied.

To copy a step or group of steps:

1. Select the step or steps to be copied.
2. Click the  button.
or
Press Ctrl+C on the keyboard.
This copies the step or steps to the Clipboard.

Related Topics

[Cut a step](#)

[Paste a step](#)

Cut a step

You can cut a step or group of steps, and then paste those copies into the same handler window or another handler or subroutine. Select a group of steps by drawing a box around them with the mouse pointer. The selected steps are individually highlighted.

To cut a step or group of steps:

1. Select the step or steps to be cut.
2. Click the  button.
or
Press Ctrl + X on the keyboard.
This cuts the step or steps to the Clipboard.

Related Topics

[Copy a step](#)

[Paste a step](#)

Paste a step

You can paste a step or steps from the Clipboard into a handler or subroutine. Any pasted steps have the same properties as the originals. Steps copied from one handler window can be pasted into the same handler window or into another handler or subroutine.

To paste a step or group of steps:

1. Select the handler window where the step or steps will be pasted.



2. Click the button.

or

Press Ctrl+V on the keyboard.

This pastes the step or steps from the Clipboard into the handler.

Related Topics

[Copy a step](#)

[Cut a step](#)

Delete a link

To delete a link:

1. Click on the link that you want to delete.
Anchor points appear along the link, letting you know that the link is selected.
2. Press the DELETE key on your keyboard.
The link is deleted.

Working with Audio Prompts

Prompt Editor

Prompt Editor is the tool you'll use to create and manage CIC Prompts. Prompts are voice or music recordings in a proprietary CIC format. Prompts are smaller than other types of audio recordings, and they are opened and played more quickly than .WAV files. These features are important for an interactive voice response system.

Choose one of the topics below for information recording, editing, and deleting Prompts:

[Record a prompt](#)

[Listen to a prompt](#)

[Delete a prompt](#)

[Localizing Prompts](#)

Note: If the audio quality is inferior, check to make sure the Microsoft CCITT G.711 and mu-law CODECS are installed. To do this, open your Microsoft Control Panel and open the Multimedia Properties. Click the Advanced Tab, and then expand the Audio Compression Codecs branch. A list of codecs appears. If you don't see Microsoft CCITT G.711 and mu-law CODECS, click Add, and then install the missing codec.

The Prompt List

The Prompt list contains a list of all the Prompts stored in Directory Services (EIC\Recordings.) Each listing has five attributes: Prompt Name, Duration, Modification Date, Description, and TDD Prompt. These attributes are described below.

Prompt Name

This is the name of the prompt set at the time it was recorded. Use this name when playing a prompt from a Play Prompt step.

Duration

This is the prompt's length in seconds.

Modification Date

This is the date on which the Prompt was last modified.

Description

This is a brief description of the prompt, created when the prompt was first recorded.

Chat Text

The text of the prompt as it will appear in a chat interaction.

TDD Text

The text of the prompt as it will appear in a TDD device.

Input Mode

Where applicable, this designates the input mode (DTMF, Voice, etc.) of the prompt. The input mode is set to Any by default.

The Prompt Buttons

This section describes the ten Prompt Editor buttons.

OK

Saves any changes you have made and closes the Prompt Editor.

Cancel

Closes the Prompt Editor without saving any changes you have made.

Insert

Opens the New Voice Prompt dialog box where you can specify the name and description of a new prompt and the TDD prompt.

Delete

Deletes a prompt you have selected from the Prompt list.

Edit

Allows you to edit an ID, description, text, and Input Mode.

Search

Allows you to search for prompts containing specific words or phrases.

Play

Plays a prompt you have selected from the prompt list.

Record

Starts recording a new prompt that you created with the Insert button.

Import .WAV

Creates a prompt from a .WAV file of any format. This tool does not support MP3 files.

Export .WAV File

Creates a .WAV file from a prompt file you selected from the prompt list.

Trim Options

The Trim Leading Silence and Trim Trailing Silence options allow you to remove any silent moments at the beginning or end of a prompt. The silent moments are trimmed when you click the stop button to end the recording of a new prompt. You cannot trim silence from a prompt created in a previous recording session.

Trim Leading Silence check box

Select this box if you want to remove the silence at the beginning of a prompt you are recording.

Trim Trailing Silence check box

Select this box if you want to remove the silence at the end of a prompt you are recording.

Record a prompt

Use this procedure to record a new CIC prompt. Before you begin, make sure that the recording properties for your operating system are configured correctly. Refer to the Windows online documentation for more information on setting up recording properties. Also make sure that your microphone is plugged in.

To record a prompt, open the handler that will play the prompt, and then open a Play Prompt step. Because there is no select button in the Prompt Editor, the prompt you have selected when you click the OK button is the prompt that appears on the Inputs page.

To record a new prompt:

1. Click Edit on the Inputs page of a Play Prompt step.
The [Prompt Editor](#) appears.
2. Click Insert.
The Prompt dialog box appears.
3. Type a name and description for the new prompt and click OK. This is the name you will specify when playing back the prompt with a [Play Prompt](#) or [Play Prompt Extended](#) step.
4. Click Record.
5. Begin speaking.
The words you speak are recorded in the prompt.
6. Stop the recording by clicking Stop.
The prompt and its duration are displayed in the Prompt Name list. The Prompt is played back automatically when recording is finished.

Listen to a prompt

To listen to a prompt:

1. Click the Edit button on the Inputs page of a Play Prompt step.
The Prompt Editor appears.
2. Select a prompt from the Prompt name list.
3. Click the Play button.
The prompt is played.

Note: If the audio quality is inferior, check to make sure the Microsoft CCITT G.711 and u-Law CODECS is installed. To do this, open your Microsoft Control Panel and open the Multimedia Properties. Click the Advanced Tab, and then expand the Audio Compression Codecs branch. A list of codecs appears. If you don't see Microsoft CCITT G.711 and u-Law CODECS, click the Add button and install the missing codec.

Delete a prompt

To delete a Prompt:

1. Click the Edit button in a step that requires an audio clip.
The Prompt Editor appears.
2. Select a prompt from the Prompt Name list.
3. Click the Delete button.
The prompt is deleted.

Importing and Exporting Prompts and Strings

These utilities are used for managing and editing all the prompts of a given handler.

Importing Prompts and Strings

This function allows you to insert your own prompts and strings into a handler. Use this function for adding custom prompts and strings, and for localization of prompt and string sets. To use this function, you must modify the four .csv (comma delimited text) files described below. These files are created automatically by the Export Prompts and Strings function. Interaction Designer will use these tables to incorporate the files properly and to add their information to the appropriate tools.

Exporting Prompts and Strings

This function exports all the prompts and strings of the active handler into a specified directory. Prompts are exported as .wav files and strings are exported as fields within the .csv files. Whenever this function is used, four .csv files (described below) are also exported. These files contain tabular information pertaining to the exported prompts and strings and can be used to facilitate customizing and/or localizing the prompt sets.

Import/Export Tables

Four .csv (comma delimited text) files are used for importing and exporting prompts and strings. These files contain the information Interaction Designer needs to properly import new prompts and strings, and they also contain all the information you need to manage exported prompts.

These files are most easily edited with MS Excel. Though they are text files, and as such can be opened and edited with many different applications, it is important to only use an editor that will maintain their format. Interaction Designer will not be able to use them properly if their formatting is altered.

[handler name]_[language code].csv

Example: Prompt_Attendant_en-US.csv

This file contains the definitions of all the prompts in the handler. From left to right, the fields in this file are:

Field	Description
Prompt File	The name of the .wav file.
Description	A transcription of the words spoken in the .wav file.
Chat Text	The text that will appear in a text session that calls this prompt.
TDD Text	The text that will appear on a TDD that calls this prompt.
Handler	The name of the handler containing this prompt.
Prompt Name	The name of the prompt as it appears in handlers.
Language	The language code describing the language in which the prompt is recorded. All default prompts are recorded in U.S. English and have a language code of en-US.

[handler name]_[language code]_SEQUENCE.csv

Example: Prompt_Attendant_en-US_SEQUENCE.csv

This file contains all the prompt-related sequence strings in this handler. From left to right, the fields in this file are:

Field	Description
Sequence String ID	The name of the sequence string.
Singular Sequence String	The sequence string as it would be used in the singular (for example, "A brown dog").
Plural Sequence String	The sequence string as it would be used in the plural (for example, "Some brown dogs").
Description	A description of the prompt.
Language	The language code describing the language in which the string is written. All default strings are written in U.S. English and have a language code of en-US.

[handler name]_ [language code]_ISTRINGS.csv

Example: Prompt_Attendant_en-US_ISTRINGS.csv

This file contains all the international strings in this handler. From left to right, the fields in this file are:

Field	Description
String ID	The name of the international string.
Description	A description of the string.
String Value	The string value for the string ID.
Language	The language code describing the language in which the string is written. All default strings are written in U.S. English and have a language code of en-US.

[handler name]_ [language code]_IS_SEQUENCE.csv

Example: Prompt_Attendant_en-US_IS_SEQUENCE.csv

This file contains all the sequence strings related to international strings in this handler. From left to right, the fields in this file are:

Field	Description
Sequence String ID	The name of the sequence string.
Singular Sequence String	The sequence string as it would be used in the singular (for example, "A brown dog").
Plural Sequence String	The sequence string as it would be used in the plural (for example, "Some brown dogs").
Description	A description of the sequence string.
Language	The language code describing the language in which the string is written. All default strings are written in U.S. English and have a language code of en-US.

Related Topics

[Assemble Prompt Phrase](#)

[Assemble Text Phrase](#)

[Localizing Prompts](#)

[Play Prompt](#)

[Play Prompt Phrase](#)

[Prompts](#)

Localizing Prompts

To create versions of the existing prompts in a different language, simply follow these steps:

1. Create a folder that will be used to hold the imported and exported prompts. You may want to create different folders for each language, however this is not required.
2. Open the handler you want to localize in Interaction Designer.
3. Select Export Prompts and Strings from the Utilities menu and export the prompts into the folder you created in step 1.

Feel free to exit out of Interaction Designer after doing this.

4. Navigate to the folder into which you exported the prompts.
5. Locate the file "[handler name]_[language code].csv". If you are using separate folders for each language, copy this file into the folder that will be used for the new prompts and open it. Otherwise, simply open the file.
6. Use the "Save As" command under the file menu to rename the .csv file to contain the language code for the new prompts. For example, if the default prompts exported from Prompts_IVR are to be re-recorded in Chilean Spanish, the file "Prompts_IVR_en-US.csv" file should be renamed, "Prompts_IVR_es-CL.csv". The language codes for all the prompts should also be changed in the appropriate fields in the .csv file.

Repeat this process for all four .csv files.

7. Translate the Description, Chat Text, and TDD Text fields into the language that the prompts will be recorded in. Make these changes in the .csv file and save the changes.
8. Change Language field to the appropriate language code. This change should be made in all four .csv files for all prompts. Save the changes in the .csv file.
9. Use the Description field of the .csv file as a script for recording the prompts in the new language. Save each recording as a .wav file with the same name as its English equivalent.
10. Place all the new .wav files in the same folder as the modified .csv file. If this is the same folder into which the prompts were exported, it is okay to overwrite the prompts that are there.
11. Open the handler from which the prompts were originally exported.
12. Select Import Prompts and Strings from the Utilities menu.
13. In the dialog box that appears, browse to the location of the localized prompts and select the .csv file there. Click Next.
14. Browse to the location of the new .wav files. Click Finish.

What is described above is the simplest method of localizing prompts. Simplest may not necessarily be best, however, and so there are a few options available to you that were not described above. For example, the locations of the .csv files and .wav files do not need to be static, nor do both file types need to be kept in the same locations. Feel free to create separate folders for each file type if you want. Also, files do not necessarily need to be exported to or imported from the hard drive. Prompt .wav files can be saved to a CD and imported directly from there, as long as Interaction Designer can access the appropriate device. This can be useful if disk space on the server is an issue.

Note: individual prompts cannot be customized by this method. The steps described can only be used to import and export the entire prompt set of a given handler.

Related Topics

[Assemble Prompt Phrase](#)

[Assemble Text Phrase](#)

[Language Codes](#)

[Play Prompt](#)

[Play Prompt Phrase](#)

[Prompts](#)

Prompt Libraries

All the prompts, prompt phrases, text strings and sequence strings that are shipped with this product are found in the following handlers. When editing the existing prompts or creating new prompts, we strongly recommend storing prompts **only** in these handlers.

Prompt_Attendant.ihd

Contains all the prompts and prompt phrases used by the Interaction Attendant handlers.

Prompt_Custom.ihd

This is an empty handler provided as a storage place for holding any custom prompts you may want to use in your own handlers or in the customization handlers.

Prompt_IVR.ihd

Contains all the prompts and prompt phrases played during IVR.

Prompt_System.ihd

Contains all prompts other than those used by the Interaction Attendant handlers and those played during IVR.

Strings_Attendant.ihd

Contains all the text strings used by the Interaction Attendant Handlers.

Strings_Custom.ihd

This is an empty handler provided as a storage place for holding any custom strings you may want to use in your own handlers or in the customization handlers.

Strings_IVR.ihd

Contains all the text strings used during IVR.

Strings_System.ihd

Contains all text strings other than those used by Interaction Attendant and those played during IVR.

Strings_Web.ihd

Contains all text strings used by the Web Services handlers.

Language Codes

Language codes are expressed as two two-letter abbreviations separated by a dash, for example, "en-US". The two parts of this code correspond to the language spoken and the country representing the specific dialect being used. In this example, "en" stands for English, and "US" for United States, thus en-US represents the dialect of English spoken in the United states. The two-letter abbreviations for both language and country and taken from those set by ISO standards.

Related Topics

[ISO Country Codes](#)

[ISO Language Codes](#)

Other reference

Retrieving the values of server and system parameters from handlers

Server and system parameters are variables whose values are set in Interaction Administrator. These values can be retrieved from handlers, and from other locations within Interaction Administrator. For example, the Held Call Timeout server parameter determines the number of seconds a call remains on hold before the Held Call Timer initiator starts a handler.

You can create your own custom server and system parameters and retrieve their values from within handlers. For example, suppose you want to create a handler that retrieves the email address of the web administrator. If you hard coded the email address in Send Email steps, you would have to modify your handlers each time you change web administrators. If the email address is stored in a server parameter, any handler can retrieve that value each time it runs. When you change the email address stored in the server parameter, all the handlers that retrieve that server parameter value are dynamically updated the next time they run.

See the Interaction Administrator online help for a list of default server and system parameters.

The difference between server and system parameters

Server parameters are available only on a particular CIC server and system parameters are available on all CIC servers on a network. In a future release of CIC that supports multiple CIC servers, system parameters will become more useful. Until that time, you can store values in either server or system parameters.

Caution: Any custom server or system parameters you create are overwritten if you perform a full install. If you are just performing an upgrade or refresh install, you will not lose your custom server and system parameters.

To create a server or system parameter:

You can create, edit, and delete server and system parameters in Interaction Administrator. See the Interaction Administrator documentation for complete instructions on configuring server and system parameters.

After you have created (and assigned a value to) a server or system parameter, you can retrieve that value from a handler.

To retrieve the value of a server or system parameter from a handler:

Server and system parameter values are retrieved with a [GetDSAttr](#) step. The GetDSAttr step requires two input fields, *Directory Services Path* and *Directory Services Attribute*. Use the following descriptions as guidelines for the values you should place in these fields when you retrieve a custom server or system parameter value.

Directory Services Path

This is the path to server or system parameter in Directory Services. In the following examples, substitute `your_server_parameter_name` with the actual name of your server or system parameter.

If you created a server parameter, type:

```
"${Server}\Parameters\your_server_parameter_name"
```

If you created a system parameter, type:

```
"${Config}\Parameters\your_system_parameter_name"
```

Directory Services Attribute

This is the name of the attribute you want to retrieve. For custom server and system parameters, you should type:

"Value"

Specify an output variable to contain the List of Attribute Value. This output will contain the value in the custom server or system parameter. Remember that this is a List of String type variable. If you need to convert the value to a string, use the [GetHead](#) operation.

Once you have saved and published the handler, you can begin retrieving the values of system and server parameters from within handlers.

Call Analysis

Call analysis is an option for almost all remote calls made from CIC clients and handlers. Call analysis determines how Telephony Services (TS) monitors outgoing calls. When call analysis is activated, TS monitors the outgoing call to see if it connected to a person or answering machine. TS will also try to diagnose why a call did not connect and display the reason in the CIC client (e.g., remote busy, no answer, etc.). If call analysis is turned off, TS doesn't monitor the call, so those calls are displayed as "Connected" or "Disconnected" in the CIC client.

Read further for more information on call analysis and when you should use it.

Where is call analysis turned on and off?

Call analysis can be turned on or off in the following locations:

- **Configuration Page in Interaction Desktop**
Set the **Analyze outgoing external calls** option on the Calls tab on the Configuration page. This option determines if the outgoing call should use call analysis. If you include an extension in your dialstring, call analysis is used automatically, even if you have explicitly turned call analysis off. Extension dialing for remote calls consists of a primary telephone number followed by a "/" character followed by an extension. For example, 872-3000/114 will dial extension 114 after 872-3000 connects.
- **Interaction Connect**
Your CIC administrator uses Interaction Administrator to configure the Analyze outgoing external calls option.
- **Extended Place Call tool in a handler**
This tool places outgoing calls from handlers, and is used in the default CIC handlers and the Interaction Dialer handlers. You can select or clear the Call Analysis option on the Inputs page for this tool. You can also select or clear Answering Machine Detection, an extended form of call analysis.

What does call analysis actually do?

When call analysis is turned *on*, TS monitors the outgoing call to see how it is proceeding through the network. This occurs on all types of lines, both analog and digital. For this process analysis, TS uses a voice resource.

TS can determine if a call has been answered, or why it wasn't answered. For calls made from a CIC client, the CIC client displays the call's status. For example, if call analysis is on and you place a call to a busy number, you will see Disconnected [Busy]. If an Extended Place Call tool makes the call to a busy number, it will take the Busy exit path. In both cases TS can determine the state of the call.

Here are some other status messages you might see with call analysis turned on:

- For busy signal, it displays "Disconnected[Remote busy]".
- For unanswered calls, it displays "Disconnected[No answer]". (You can configure how long TS waits before displaying this status in the Calls configuration page.)
- For SIT tones, it displays "Connected" and allows you to hear the SIT tone and message.

(SIT means Special Information Tone. For example, when a number has been disconnected, you hear the SIT tone, and then "The number you have dialed, 123-4567, has been disconnected or is no longer in service. Please check the number and try your call again.")

- For answered calls, it displays "Connected".

Another feature of call analysis is voice detection. TS monitors the line for a voice at the other end of a connected call to determine if a party answered. If TS does not detect a voice, it disconnects the call. That's why calls to some automated services, such as paging services that play only a tone, can cause an analyzed call to disconnect.

What is answering machine detection?

The Extended Place Call tool allows you to extend call analysis with answering machine detection. TS continues to listen over the allocated voice resource for a short or long greeting. Short greetings like "Hello" are presumed to be human. Longer greetings like "Hi, this is the Jones Residence. We can't take your call right now..." are presumed to be answering machines. Keep in mind that answering machine detection takes longer, so the outbound calls from Extended Place Call will take longer to evaluate before taking an exit path.

Answering machine detection is also available for outbound calls that include an extension (a "/" character followed by digits). TS looks at the value of the *Extension Dialing Analysis Type* server parameter. The value of this server parameter can be either "Answering Machine" or "Voice".

With Extension Dialing Analysis Type set to "Voice", CIC waits for the dialed number to be answered by either a person or machine and then automatically sends the DTMF tones for the extension. This has the advantage that you can always be certain that the

DTMF will be sent. The disadvantage is that when a human answers the call, the DTMF will play in that person's ear. Interaction Dialer has additional features for this analysis type. In that application, agent extensions are dialed even if an answering machine is detected.

With Extension Dialing Analysis Type set to "Answering Machine", CIC attempts to recognize an IVR or automated voice before sending the DTMF tones. If it does not detect an automated voice, it assumes a human voice answered and does not send the DTMF extension digits. This kind of detection is less reliable than voice detection.

Is call analysis different on digital lines?

Yes, but only slightly. For digital lines, information about the call is sometimes returned in a D-channel message for ISDN and in signaling bits for T1 and E1. These messages might be that the call is connected or that the line is busy. If TS receives information that the call is connected and voice call analysis is used, then the voice resource is no longer needed to determine if the call connected. That's why you may never hear a busy signal on a call placed over an ISDN line. You will only see "Disconnected [Busy]."

If answering machine call analysis is used, then the voice resource is still needed to evaluate the answer even though a digital indication was received that the remote party answered the phone.

What happens if I turn Call Analysis Off?

If you turn call analysis off, TS will not analyze outgoing call or monitor the line for a voice. In a CIC client, all outgoing calls are immediately displayed as "Connected", even if the remote party is busy. Sometimes this is desirable, such as when you call a paging service that plays only a tone.

If you turn off call analysis in the Extended Place Call tool, the tool may only use the Success and Failure exit paths. (In some cases, if the tool is placing calls over a digital line, the D-Channel message may be enough to take one of the other exit paths.) See the Extended Place Call tool for more information.

Note: Again, there is an exception with ISDN lines. On ISDN lines, the call will disconnect if the ISDN Network sends TS a busy indication.

Object States

States

As call objects (calls) move through the CIC system, the CIC hardware and software assign states to those calls. States have two purposes in CIC. First, they show CIC client users the condition of calls in a queue. A CIC client user can tell if a call is On Hold, Disconnected, or any other state just by looking at the icon associated with a call.

The second purpose of states is to define for the CIC system what operations can be performed on a call. For example, a call in an "On Hold" state cannot be placed in a "Dialing" state. Also, some tool steps only perform actions on call objects that are in a certain state. For example, the Record Audio step can only record a call if it is in a "Connected" state.

Valid call states include:

- **Alerting** (seen for inbound calls)
A CIC client user is being notified (through ringing) that he or she has an incoming call.
- **Connected** (seen for both outbound and inbound calls)
Both parties are connected and are able to speak with each other. If [call analysis](#) has not been enabled in the CIC client, Connected means the same as Proceeding.
- **On Hold** (seen for both outbound and inbound calls)The call is on hold.
- **Voice Mail**
The call is leaving a voice mail message. While Voice Mail appears as a state to CIC client users, Voice Mail is not really a state. Calls that appear in a state of Voice Mail are actually in a state of Connected. See the *Interaction Attributes Reference Guide* for information on displaying custom strings to CIC client users.
- **Offering** (seen for inbound calls)
The call has been placed in a queue, but the call is not ringing. CIC is determining if the called party is available to take the call.
- **Parked**
The call has been placed on hold on the recipient's My Interactions queue.
- **Proceeding** (seen for outbound calls)
The call is proceeding through the outside telephone network. Proceeding is used if an CIC client user has enabled [call analysis](#).
- **System**
The call is interacting with the xIC system (IVR, etc.).
- **Disconnected Locally**
The call was disconnected by the local participant.
- **Disconnected by Remote Party**
The call was disconnected by the remote participant.

Note: There are many different reasons for call disconnects. For a reference of ISDN Cause Codes related to disconnected calls, see the ITU-T Q.850 specification.

The state display string is maintained in the Eic_CallStateString attribute. It is used to construct the string displayed in CIC client in the column entitled State. This display string can be constructed as a localizable string, meaning that certain numeric values enclosed by % characters will be replaced with text strings on the client when the value is displayed. For example, this allows a Client running on a Spanish PC to display the Spanish text for "Disconnected" while a German client displaying the same call would show the German text for "Disconnected." The numeric values used by Queue Manager are shown below:

Localizable Display

String String

%13664% Connected

%13665% Held

%13666% Voice Mail

%13667% Parked

%13668% System

%13669% Disconnected (Local Disconnect)

%13670% Disconnected (Remote Disconnect)

%13671% Proceeding

%13672% Offering

Tools impacted by the PureConnect data privacy feature

Customers can prevent potentially sensitive data from appearing in trace logs. The PureConnect data privacy feature can suppress the trace logging of data that might be sensitive. This feature is enabled by setting two server parameters. For background information, see PureConnect data privacy feature, in Chapter 6 of the [Security Precautions Technical Reference](#). To access this secure content, you must login to the Genesys Product Information website.

IP Tool Tracing Changes

Many IP tools access and trace data that may be considered sensitive. When the SuppressSensitiveDataTracing server parameter is enabled, the input/output values in the table below are suppressed. The following tools will trace ##Suppressed## for potentially sensitive data if the SuppressSensitiveDataTracing server parameter is set to "True".

Tool Name	Suppressed Values
Array Builder	Values List, Array String
Array Parser	Array String, Values List
Filter List	String to be used for filter, List of strings to be processed
Find	Value
InsertAt	Value
InsertAtHead	Value
InsertAtTail	Value
JSON Builder	Values List, JSON string
JSON Parser	JSON string, Values List
ListToString	List of Strings, Delimited String
LookUp	Attribute Value
Lookup List	Attribute Value
Lookup List Extended	Attribute Value
Merge String Lists	Source List 1, Source List 2, Merged List
Parse String	String to Parse, List of Parsed Strings
Parse String RegEx	String, Result
RegEx Extended	String, Result, Remainder
REST HTTP Request	Raw Request Body, Response Body
SetAt	Value
SOAP Add Body Element	Value
SOAP Add Header Element	Value
SOAP Add RPC Parameter	Value
SOAP Base64 Decode	Encoded Data, Decoded Data
SOAP Base64 Decode To File	Encoded Data
SOAP Base64 Encode	Data, Encoded Data
SOAP Base64 Encode File	Encoded Data

SOAP Create Array	Values
SOAP Create RPC Response	Return Value
SOAP Get Body Element	Value
SOAP Get Fault	Fault String
SOAP Get Header Element	Value
SOAP Get Next RPC Parameter	Value
SOAP Get RPC Parameter	Value
SOAP HTTP Request	Raw Response Body
SOAP HTTP Request Ex	Raw Response Body
SOAP HTTP Request Ex2	Raw Response Body
SOAP HTTP Request Ex3	Raw Response Body
Sort Lists	List of strings to be processed, Parallel list(s)
TCP Read String	String value
TCP Read String UTF8	String value
TCP Write String	String value
TCP Write String UTF8	String value
Update Data Pair Values	DataValue, Value List

Interaction Designer interface

Introduction to the Design Palette

The Design palette contains all the tools and subroutines available for building handlers. The Design palette is divided into two pages, one containing all the available tools, and one containing all available subroutines.

You can reposition the Design palette as desired by clicking on its title bar and dragging and dropping it in the desired location. You can resize the palette by clicking on a corner and dragging. And, you can close the Design palette by clicking the Close button on its title bar. To make the palette visible again after it has been closed, click Design palette on the View menu.

The Tools Page

The Tools page contains all of the tools available for building handlers. Each tab on the tool page contains a group of related tools. For example, the Database tab displays all tools for opening, closing, and accessing databases.

Note: You can configure which tools appear on the Tools page of the Design palette by modifying the file DesignerRegisteredTools.lst. This ASCII file contains a list of the .DLL files that define the tools. Placing a '#' sign in front of one of the .DLLs will cause that group of tools to not appear on the Tools page. When Interaction Designer starts, it first looks for DesignerRegisteredTools.lst in the current Interaction Designer directory and then in the system path. The first DesignerRegisteredTools.lst file found is the one that is used.

If you open a handler that contains a tool that is not loaded in Interaction Designer, that tool appears in the handler without an icon. That tool can be configured normally and will compile normally when you publish the handler.

The Subroutines Page

The Subroutines page contains all of the tools for calling subroutines in handlers. Each tab on the Subroutines page contains a group of related subroutine tools. The groupings and labels for the subroutine tools are created when subroutines are **published**. You can create a new subroutine category and a new subroutine tool when you publish a subroutine. Once a subroutine has been published, its tool will appear in the Subroutines page.

Related Topics

[Drag a tool from the Tools page of the Design palette to create a step Tools](#)

Disabled Features in Interaction Designer

This topic describes menu items that are dependent on the server connection.

Features not available without a server connection

When Interaction Designer is not connected to a server, the following menu items are disabled in Interaction Designer:

File | Download From Server

File | Publish

File | Debug Immediate

Utilities | Manage Handlers

Utilities | Debug Handlers

Utilities | Stored Procedures

Utilities | View Dependencies

Note: When menu items are disabled due to not having a Notifier connection, "No Notifier Connection" appears next to the menu text.

File Menu Commands

The File menu offers the following commands:

Command	Description
New	Creates a new document.
Open	Opens an existing document.
Close	Closes an opened document.
Save	Saves an opened document using the same file name.
Save As	Saves an opened document to a specified file name.
Download Handler From Server ...	Downloads one or more handlers from the server.
Print	Prints a document.
Print Preview	Displays the document on the screen as it would appear printed.
Print Setup	Selects a printer and printer connection.
Properties	Opens a box where a description of the handler can be entered.
Export To XML	Creates an XML document containing all the information in the current handler.
Change Initiator	Changes the Initiator for the current handler, or allows you to select an initiator for a new handler.
Publish	Publishes a handler to the CIC server.
Debug Immediate	launches a debugging session for the currently published version of the active handler.
Generate .i3pub File	Creates an intermediate publish file in a specified directory.
Import Global Variables	Imports global variables contained on the CIC server.
Exit	Exits Interaction Designer.

Related Topics

[Disabled Features in Interaction Designer](#)

Edit Menu Commands

The Edit menu offers the following commands:

Command	Description
Cut	Deletes data from the document and moves it to the Clipboard.
Copy	Copies data from the document to the Clipboard.
Paste	Pastes data from the Clipboard into the document.
Select All	Selects all steps in the active handler window.
Preferences	Opens the Designer Preferences page.

View Menu Commands

The View menu offers the following commands:

Command	Description
Layout Toolbar	Shows or hides the Layout toolbar.
Palette Toolbar	Shows or hides the Palette toolbar.
Standard Toolbar	Shows or hides the standard toolbar.
Debug Palette	Shows or hides the Debug palette.
Design Palette	Shows or hides the Design palette.
Messages Palette	Shows or hides the Messages palette.
QuickJump Palette	Shows or hides the QuickJump palette.
Variables Palette	Shows or hides the Variable palette.
Status Bar	Shows or hides the Status bar.

Layout Menu Commands

The Layout menu offers the following commands:

Command	Description
Grid	Toggles the background grid in the design window.
Go	Scrolls the design window up, down, left or right.
Zoom	Controls the zoom level in the design window.
Align Selected Nodes	Aligns the selected steps along the designated border of the primary selected step.

Utilities Menu Commands

The Utilities menu offers the following commands, which perform useful actions in Interaction Designer.

Command	Description
QuickJump	Opens the QuickJump Palette to allow you to search for a step in a handler or subroutine.
Manage Handlers	Opens the Manage Handler notebook.
Debug Handlers	Opens the Debug a Handler dialog box where you select a handler to debug.
Stored Procedures	Opens the Stored Procedures Definition dialog where you can modify a stored procedure definition.
View Dependencies	Opens the Dependencies notebook.
SOAP Wizard	Builds a SOAP request subroutine you can use with the SOAP tools. You define the calling subroutine and the variables to use.
Power Tools	Opens a list of extensions to standard Interaction Designer functionality. These tools are intended for veteran handler developers.
Import Prompts and Strings	Opens a dialog that allows you to import a list of .wav files and their transcripts.
Export Prompts and Strings	Exports the prompts within the open handler. Each prompt is exported as a separate .wav file.

Related Topics

[Disabled Features in Interaction Designer](#)

Debug Menu Commands

Command	Description
Re-Start Debug Session	Restarts the debugging session with all the same settings (breakpoints, viewed variables, etc.).
Run	Causes a handler to run until it reaches the next break point or until the session ends.
Stop	Stops the debug session. When the debugging session is stopped by this method, the user will be prompted if they want to allow the handler to continue when the debugging session ends.
Step	Moves to the next step in the handler. This is only used when single-stepping through handlers.
Step In	Enters a subroutine in the debugging session. This is only available when the handler is paused on a subroutine step.
Step Out	Steps out of a subroutine and returns to the calling handler.

Window Menu Commands

The Window menu offers the following commands, which enable you to arrange multiple views of multiple documents in the application window:

Command	Description
New Window	Creates a new window that views the same document.
Cascade	Arranges windows in an overlapped fashion.
Tile	Arranges windows in non-overlapped tiles.
Arrange Icons	Arranges icons of closed windows.
Handler 1, 2, ...	Goes to specified handler window.

Help Menu Commands

The Help menu offers the following commands, which provide you assistance with this application:

Command	Description
Help Topics	Opens this online help file.
About	This dialog displays the version number of this application and provides buttons to display CIC server Information and System Information for the hardware and system software.
Contact Genesys	This menu provides four links to PureConnect customer resources: Home - links to the Genesys home page Support - links to the Genesys.Support Services page, which requires a login name and password. Community - links to the Community.inin.com web site, which provides a place for CIC users to post questions and share information. Feedback - links to a PureConnect product feedback form where you can send key people direct product feedback.

Interaction Designer's Help menu also allows you to add your own documentation and/or help files for any custom handlers you may have created. To add an item to Designer's Help menu, simply place the file or directory in the folder ...\\13\\Server\\Help. The next time Interaction Designer is started, anything located in that directory will appear in the Help menu beneath the "Help Topics," "About," and "Contact Genesys" options.

Toolbars



Interaction Designer has three toolbars that can be displayed across the top of the application window, below the menu bar. The toolbars provide quick mouse access to many tools used in Interaction Designer,

To hide or display each toolbar, choose the appropriate toolbar from the View menu.

[Layout Toolbar](#)

[Palette Toolbar](#)

[Standard Toolbar](#)

Designer Preferences Dialog Box

The Designer Preferences dialog box allows you to set a number of default behaviors in Interaction Designer. ID will use these settings automatically whenever appropriate. If a preference is not set in this page, ID will prompt the user for the information each time the relevant action is performed.

To display the Designer Preferences dialog box, click the Edit menu, click Preferences, and then click Designer Preferences.

The Designer Preferences dialog box contains four tabs:

[Debugging](#)

[General](#)

[Handler Download](#)

[Views](#)

Expressions, Data Types, and Operators

Creating Expressions

Expressions are formulas that process [literal values](#), [variables](#), and [operators](#) to create a new value. You'll find this type of processing useful in all but the simplest handlers, so understanding how to create expressions will allow you to create flexible and powerful handlers. This topic offers the information you need to begin building your own expressions.

You can create an expression for any input parameter that takes a [normal value](#). You cannot create expressions for input parameters that take [extended values](#), such as Call ID. Also, you cannot create an expression in an output parameter. Output parameters can contain only variables that receive a tool's output values.

See [Expression Editor Assistant](#) for detailed instructions on creating expressions.

Sample Expressions

Maximum Number of Keys

Expression: 1

The maximum number of keys a caller may press before this tool exits is 1.

Escape Keys

Expression: "*"

The valid string values for Escape Key is the * key. (Since this input parameter takes a string, the * is within quotation marks.)

Call ID

Expression: CallA

The identifier for this call is stored in the CallA variable.

User Queue

Expression: StrMid ("User Queue:StephenS",11,30)

Returns 30 characters starting at character 11, effectively removing the "User Queue:" portion of the specified string.

Prompt Name

Expression: "IVR_REMOTE_VM_" & PromptName

Combines two strings into a single string; one is hard coded and one is contained in the PromptName string variable.

Agent Name

Expression: GetHead(AvailableAgentList)

Returns the first string in a list of strings.

Variable: Loopcount

Expression: Loopcount + 1

Increments the integer variable loopcount by 1.

Path

Expression: "c:\\root\\foos"

When CIC parses this string, the \\ is interpreted as a single \. This is explained in the Expression Editor Assistant section.

Related Topics

[Creating Expressions Without Using Expression Editor Assistant](#)

[Expression Editor Assistant](#)

[Limiting Available Literal Values, Variables and Operators](#)

[Learning Expression Editor Assistant Quickly](#)

Literal Values, Variables, and Operators

Expressions are composed of at least one of the following elements:

- [Literal Values](#)
- [Variables](#)
- [Operators](#)

Tool input parameters expect data of a specific type. There are three ways to enter that value into the parameter:

- You can type a literal value, such as "User Queue: Stephen Schiller" or 1970.
- You can specify a variable such as StrUserQueue or IntBirthYear.
- You can build an expression with Operators that produce the desired value, such as GetYear (dtBirthyear)

See [Creating Expressions](#) for information on the purpose of expressions in Interaction Designer.

Expression Editor Assistant

Expression Editor Assistant

The Expression Editor Assistant helps you write expressions by checking for errors as you type. It also assists by completing simple tasks for you. For example, if you type a quotation mark (to specify a string value), it inserts the other quotation mark for you. This ensures that your string value is contained within quotation marks. In another example, if you begin to type an operator, Expression Editor Assistant will recognize the operator, finish typing it, and insert "?" symbols for missing [operands](#). This type of assistance and auto-correction can be a little unnerving, but it's helpful when you become proficient writing expressions. It also ensures correct syntax.

Related Topics

[Creating Expressions](#)

[Limiting Available Literal Values, Variables and Operators](#)

[Learning Expression Editor Assistant Quickly](#)

[Creating Expressions Without Using Expression Editor Assistant](#)

Limiting Available Literal Values, Variables, and Operators

One important concept to remember about Expression Editor Assistant is how it limits the available [literal values](#), [variables](#), and [operators](#) to those that are valid for the expected type. For example, if you open Expression Editor Assistant for a string input parameter, only string literal values, string variables, and operators that produce strings are available. Similarly, when you have selected an [operand](#) in an operator, the only available literal values, variables, and operators are those that are valid for that operand type.

Related Topics

[Expression Editor Assistant](#)

[Creating Expressions](#)

[Learning Expression Editor Assistant Quickly](#)

[Creating Expressions Without Using Expression Editor Assistant](#)

Learning Expression Editor Assistant Quickly

The following list contains some sample expressions. Type, key for key, the text following **Type:** to achieve the expression shown after **To get:**. Watch closely how Expression Editor Assistant inserts and completes the syntax of your expression. This list was designed to show you most types of assistance.

Type: "dog
To get: "dog"

Expression Editor Assistant adds the 2nd quotation mark when you type the first.

Type: strlen("dog
To get: StrLen("dog")

Expression Editor Assistant adds the 2nd parenthesis and quotation mark

Type: Strm(
To get: StrMid(?, ?, ?)

Expression Editor Assistant recognizes and completes StrMid and inserts question marks for the operands, and adds the 2nd parenthesis.

Type: \
To get: "\\

Expression Editor Assistant changes the \ to a \\. The \ symbol is interpreted as a flag for a special string character such as \t (tab) or \n (new line). \\ is a flag that is interpreted as a single \ when the tool executes.

Type: "c:\\root\\foos
To get:"c:\\root\\foos"

When this path is parsed, the \\ is interpreted as a single \, resulting in "c:\root\foos".

Note: When you type the first \, Expression Editor Assistant inserts the second \ character automatically. If you want to keep the second \ character, just type the second \ character. Expression Editor Assistant understands that when you type \\ you want both. If you do not want the second \ character, typing any character other than a \ removes the second \. Try this for yourself in an Assignment step in an empty handler to practice.

See [Literal Values](#) operators for other values that use the \ character.

Related Topics

[Expression Editor Assistant](#)

[Creating Expressions](#)

[Limiting Available Literal Values, Variables and Operators](#)

[Creating Expressions Without Using Expression Editor Assistant](#)

Creating Expressions Without Using Expression Editor Assistant

You can copy a string created in a text editor (like Notepad.exe) and paste it into an expression. This can be more difficult to debug once you've pasted it into the expression field. You won't receive assistance as you type using this method, but Expression Editor Assistant still requires a valid expression before it will allow you to close the tool's properties notebook.

Related Topics

[Expression Editor Assistant](#)

[Creating Expressions](#)

[Limiting Available Literal Values, Variables and Operators](#)

[Learning Expression Editor Assistant Quickly](#)

Data types

Data Types

Each input parameter in a tool expects data of a certain type. An input parameter that expects a value of type integer cannot accept a string. An input parameter that expects a value of type string cannot accept an expression that results in a value of type Boolean. While there are many different data types in CIC, they can all be classified as either **normal** or **extended**.

Output parameters return data of a specific type. All output parameters place their data in a variable of a specific data type. For example, the Digits output parameter places a string value in the Digits string variable.

See [Expression Editor Assistant](#) for more information on creating expressions.

Normal Data Types

Normal data types can be used in expressions, can be written to and read from a database, and can be displayed in an HTML form. For example, integer is a normal type. Integer values can be used in expressions, can be read from and written to databases, and can be displayed in an HTML document.

Normal data types include:

- **Boolean**

Boolean values can have a value of either true or false. For example, the expression $4 < 5$ results in a value of true. The expression $5 < 4$ results in a value of false. When a parameter expects a Boolean value, you can assign that parameter a literal value of true or false, build an expression that results in a value of true or false, or type the name of a Boolean variable that is assigned a value in some other step. See Comparison operators for examples of Boolean expressions.

The [Comparison operators](#) produce Boolean values. Boolean values may also be typed as [literal Boolean Values](#) (true and false).

- **Date/Time**

Date/Time values contain information about a specific date and a time. The information contained in a Date/Time value is the year, month, day, hour, minute, and second. See Date/Time operators for examples of how Date/Time values can be used and modified in expressions.

The [Date/Time operators](#) process and produce Date/Time values.

- **Integer**

Integer values can be any positive or negative non-decimal numbers including zero. They include all numbers in the Whole number set. When a parameter expects an integer value, you can assign that parameter a literal value (like -1, 0, or 434), build an expression that results in an integer value, or type the name of an Integer variable that is assigned a value in some other step.

Note:

The largest positive integer value is 2,147,483,647. The smallest negative integer value is -2,147,483,648. Values outside this range cannot be entered as literal values. Adding/subtracting integers such that the result is outside this range can cause wraparound. Thus, the following equations are true at runtime:

$2,147,483,647 + 5 = -2,147,483,644$

$-2,147,483,647 + -5 = 2,147,483,644$

The [Mathematical operators](#) and the [Type Conversion](#) operators process and produce integer values.

- **Numeric**

Numeric values include all real numbers in the range -2,147,483,648 to 2,147,483,647. When a parameter expects a Numeric value, you can assign that parameter a literal value (like -1.5, 0, or 45.9348), build an expression that results in a numeric value, or type the name of a numeric variable that is assigned a value in some other step. See Operator Descriptions for examples of operations that result in numeric values.

The [Mathematical operators](#) and the [Type Conversion](#) operators process and produce numeric values.

- **String**

String values can be any sequence of text characters. When a parameter expects a string value, you can assign that parameter a literal value (like "Dog", "Turk182", or "34.56"), build an expression that results in a string value, or type the name of a string variable that is assigned a value in some other step. See Operator Descriptions for examples of operations that result in string values.

The [String operators](#) process and produce integer values.

- **Lists of (any normal or extended type)**

Lists contain one or more values of any normal or extended type. A List of strings variable might contain many string values. Each position in the list has a number, and the first position in a list is always position zero.

Note:

A list cannot contain another list; in other words, you cannot have a list of lists.

The [List operators](#) and [List tools](#) both process and produce List of values.

Extended Data Types

Extended data types can be thought of as handles for values that are foreign to Interaction Designer. Extended values cannot be used in expressions, cannot be read from or written to a database, and cannot be displayed in an HTML document.

Note:

Extended data types only appear as variables in the handlers. Input parameters that take extended types don't offer Expression Editor Assistant. Rather, they offer a drop-down list where you can choose from a list of all variables of the extended value type.

Extended values hold information that is useful for hardware and software outside the base CIC system. These external types are useful because they extend the functionality of the CIC system. New external types can even be added to the CIC system to allow CIC to integrate with non-CIC hardware and software.

Here are some of the extended data types you'll encounter in handlers. You will probably encounter others, and new types are created with each CIC release:

- **Call Identifier variable**

An identifier for a call object created by the Telephony Services subsystem.

Note:

You can convert a call ID to an integer value with the [Call ID to Integer tool](#), and an integer value to a Call ID with the [Integer to Call ID tool](#).

- **Database Connection variable**

DBConnection values contain a handle (or identifier) for a database connection. When a [DB GetConnection](#) step connects to a database, the result (or output) of that step is a handle for the connection that is stored in a Database Connection variable. Several [database tools](#) use a Database Connection value to read from and write to databases. Because DBConnection values are extended data types, they cannot be modified in expressions.

- **Database variable**

Database variables contain a handle (or identifier) for a database. When a [DBOpen](#) step opens a database, the result of that step is a handle for the opened database that is stored in a Database variable. This Database value is later used by a [DBGetConnection](#) step to establish a connection with the database. Because Database values are extended data types, they cannot be modified in expressions.

- **Fax ID variable**

An identifier for a fax object (which contains a fax) created by the [Create Fax](#) tool.

- **File Handle variable**

An identifier for a file opened with the [File Open Read](#) or [File open Write](#) tool.

- **Folder ID variable**

An identifier for an opened folder in an email application, such as Microsoft Exchange or IBM Notes/Domino. See [Email tools](#) for more information on email tools.

- **Hconn variable**

An identifier for a connection with an MQSeries application created with the MQConnect tool.

- **Host Connect variable**

An identifier for a mainframe or AS400 connection created by the Host Connect tool. See [Overview of Host Interface Tools](#) for more information on Host Interface tools.

- **Socket Handle variable**

An identifier for a TCP/IP connection created with the TCP Connect Tool. See [TCP/IP tools overview](#) for more information on TCP/IP tools.

- **Web Connection variable**

WebConnection values contain a handle (or identifier) for a World Wide Web connection. When a GenerateHTML step creates an HTML document, that document is sent back to a person browsing the web via the value of the WebConnection variable. The WebConnection value is also used by several chat tools. Because WebConnection values are extended data types, they cannot be modified in expressions.

Normal Data Types

Extended Data Types

Operators

Operators are reserved words in expressions that act on literal values and variables. For each [normal data type](#), there is a set of operators that produce that data type. For example, mathematical operators perform addition, subtraction, multiplication, and other operations on integer and numeric values. String operators process strings, and so on.

Expression Editor Assistant limits your choice of operators to those that are legal for the input parameter or selected operand. For example, if you are editing an input parameter that expects a string, then only operators that result in strings are displayed in Expression Editor Assistant. This rule also applies to [operands](#). If you select an operand that expects an integer, Expression Editor Assistant limits your operator choices to those that produce an integer.

Operator Categories

For each normal value type, there are several operators that produce a value of that type. Click on one of the links below for a list of operators available for that type.

[Comparison operations](#)

[Date/Time operations](#)

[List operations](#)

[Literal values](#)

[Mathematical operations](#)

[String operations](#)

[Type Conversions](#)

Operands (Arguments)

Just like tools in a handler have parameters, many operators take arguments, or operands. For example, the operator for addition takes two input operands: ? + ?. Addition takes two integer or two numeric values for operands. When you insert an operator into an expression, the unfilled operands are represented with a "?" character.

Nested Operators

While you can use several assignment steps to construct complex expressions, you can also build complex expressions with nested operators. For example, `StrLen ("User Queue: " & QueueNameStr)` returns the length of "User Queue: " combined with the string contained in QueueNameStr variable.

CIC resolves the most deeply nested (innermost) operator(s) first, then works out to the least, or highest level operator. For example, in the following expression, **red** would be resolved first, then **blue**, then **green**.

StrLen (I>S(5*5) & UserName)

Order of Operations

In general, all arguments to functions are fully evaluated before the function is called, and all operands to operators are evaluated before the operator is applied. Exceptions are the **And** and **Or** operators, and the **Test** function, which may not evaluate all of their arguments. The order of evaluation for "sibling" subexpressions occurs left to right.

Related Topics

[Literal Values](#)

[Literal Values, Variables, and Operators](#)

[Variables](#)

Operators

Operators are reserved words in expressions that act on literal values and variables. For each [normal data type](#), there is a set of operators that produce that data type. For example, mathematical operators perform addition, subtraction, multiplication, and other operations on integer and numeric values. String operators process strings, and so on.

Expression Editor Assistant limits your choice of operators to those that are legal for the input parameter or selected operand. For example, if you are editing an input parameter that expects a string, then only operators that result in strings are displayed in Expression Editor Assistant. This rule also applies to [operands](#). If you select an operand that expects an integer, Expression Editor Assistant limits your operator choices to those that produce an integer.

Operator Categories

For each normal value type, there are several operators that produce a value of that type. Click on one of the links below for a list of operators available for that type.

[Comparison operations](#)

[Date/Time operations](#)

[List operations](#)

[Literal values](#)

[Mathematical operations](#)

[String operations](#)

[Type Conversions](#)

Operands (Arguments)

Just like tools in a handler have parameters, many operators take arguments, or operands. For example, the operator for addition takes two input operands: ? + ?. Addition takes two integer or two numeric values for operands. When you insert an operator into an expression, the unfilled operands are represented with a "?" character.

Nested Operators

While you can use several assignment steps to construct complex expressions, you can also build complex expressions with nested operators. For example, StrLen ("User Queue: " & QueueNameStr) returns the length of "User Queue: " combined with the string contained in QueueNameStr variable.

CIC resolves the most deeply nested (innermost) operator(s) first, then works out to the least, or highest level operator. For example, in the following expression, **red** would be resolved first, then **blue**, then **green**.

StrLen (I>S(5*5) & UserName)

Order of Operations

In general, all arguments to functions are fully evaluated before the function is called, and all operands to operators are evaluated before the operator is applied. Exceptions are the **And** and **Or** operators, and the **Test** function, which may not evaluate all of their arguments. The order of evaluation for "sibling" subexpressions occurs left to right.

Related Topics

[Literal Values](#)

[Literal Values, Variables, and Operators](#)

[Variables](#)

Comparison Operators

The following operators compare two values and return a Boolean value (true or false.) Some of these operators will compare alpha-numeric characters. For these operators, comparisons are made alphabetically. For example, "a" is less than "c" because a appears in the alphabet before c. Also, all lower-case letters are less than upper-case letters. For example upper-case "A" is greater than lower-case "z".

=

`(ANY_SIMPLE_TYPE) = (ANY_SIMPLE_TYPE)`

The value on the left equals the value on the right, this operator will result in a value of true.

Example: In the expression `5=5`, the result would be true because both values are equal.

In the expression `"Fred"="Barny"`, the result would be false because the string "Fred" does not equal the string "Barny".

>

`(ANY_SIMPLE_TYPE) > (ANY_SIMPLE_TYPE)`

If the value on the left is greater than the value on the right, this operator will result in a value of true.

Example: In the expression `(10>1)`, the result would be true because ten is greater than one.

In the expression `(4>5)`, the result would be false because four is not greater than five.

>=

`(ANY_SIMPLE_TYPE) >= (ANY_SIMPLE_TYPE)`

If the value on the left is greater than or equal to the value on the right, this operator will result in a value of true.

Example: In the expression `(10>=1)`, the result would be true because ten is greater than one.

In the expression `(1>=1)`, the result would be true because one is equal to one.

In the expression `(4>=5)`, the result would be false because four is not greater than or equal to five.

<

`(ANY_SIMPLE_TYPE) < (ANY_SIMPLE_TYPE)`

If the value on the left is less than the value on the right, this operator will result in a value of true.

Example: In the expression `(1<10)`, the result would be true because one is less than ten.

In the expression `(5<4)`, the result would be false because five is not less than four.

<=

`(ANY_SIMPLE_TYPE) <= (ANY_SIMPLE_TYPE)`

If the value on the left is less than or equal to the value on the right, this operator will result in a value of true.

Example: In the expression `(1<=10)`, the result would be true because one is less than ten.

In the expression `(1<=1)`, the result would be true because one is equal to one.

In the expression `(5<=4)`, the result would be false because five is not less than or equal to five.

<>

`(ANY_SIMPLE_TYPE) <> (ANY_SIMPLE_TYPE)`

If the value on the left is not equal to the value on the right, this operator will result in a value of true.

Example: In the expression `(1<>10)`, the result would be true because one is not equal to ten.

In the expression `(1<>1)`, the result would be false because one is equal to one.

Not

Not (Boolean)

Reverses the value of a Boolean expression.

Example: If the value is (4<3) is false, the value of Not(4<3) is true.
If the value is (1<2) is true, the value of Not(1<2) is false.

And

```
(Boolean) And (Boolean)  
output( BOOLEAN )
```

If the value of both operands is true, this operator will result in a value of true.

Example: In the expression (4<5) And (5<6), the result would be true because both values are true.

In the expression (4<5) And (8<7), the result would be false because at least one value is false.

In the expression (10<9) And (2<1), the result would be false because at least one value is false.

Or

```
(Boolean) Or (Boolean)
```

If the value of either operand is true, this operator will result in a value of true.

Example: In the expression (4<5) Or (6<5), the result would be true because at least one value is true.

In the expression (4<5) Or (7<8), the result would be true because at least one value is true.

In the expression (10<9) Or (2<1), the result would be false because neither value is true.

InStr

```
InStr (String, String)
```

If the value of the first string is found in the value of the second string, this operator will result in a value of true. The string comparison is case sensitive. To make the comparison case insensitive, use the [StrLower\(\) operator](#) or [StrUpper\(\) operator](#) with both strings.

Example: InStr("Dog", "Dogs") would be true because the first string can be found in the second string.

InStr("Cat", "Carbine") would be false because the first string cannot be found in the second string.

InStr("y", "XyZ") would be true because the first string can be found in the second string.

InStr("y", "xYz") would be false because the first string contains a lowercase y and the second string contains an uppercase Y.

Date/Time operators

Date/Time Functions

Using Time Zones (TZ) versus Universal Coordinated Time (UTC)

Most of the various date/time functions come in two flavors: time zone (TZ) and Universal Coordinated Time (UTC). UTC, formerly known as Greenwich Mean Time, or GMT, is the same regardless of where you are in the world or whether or not it's Daylight Saving's Time. Time Zone time, on the other hand, differs depending on how far you are from zero longitude and possibly also on what time of year it is. Obviously, calculating the datetime value based on time zones is more complicated than using UTC. This is why all datetime values in CIC are UTC times, and we strongly recommend keeping this standard when creating your handlers.

However, TZ functions are available to you if you want to base your datetime operators on time zone time instead. In the datetime parameters, "TZ" designates the timezone string parameter. These names correspond either to the time zones defined by Windows or to ICU time zone values that are specified in Interaction Administrator configuration.

Some examples of Windows time zones are: Alaskan Standard Time, Arabian Standard Time, Canada Central Standard Time, Central America Standard Time, and Dateline Standard Time. Check the following registry key to get a list of time zones supported on your server:

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Time Zones
```

Interaction Administrator, Interaction Desktop, and other systems follow the ICU standard for time zones written as configuration information. For more information, refer to <http://userguide.icu-project.org/datetime/timezone>.

If a timezone (TZ) function is left blank, the timezone will default to the local time of the CIC Server that the handler is running on. However, if a handler author wants to use a non-blank timezone string in a TZ datetime expression editor function, they need to realize that the valid timezone strings can (and will) change on a per server basis.

Note: valid TZ values are defined either by the server or by the Interaction Administrator time zone configuration. CIC reads what is there, but it does not define any values itself. For this reason, the values defined on one server may not be the same as those defined on another. To determine what time zones are defined on your server, you must read the values from the registry.

Using UTC-based datetimes has three advantages over using time zone based datetimes:

- No calculations are required to convert time from place to place.
- No possibility of error due to differing definitions of time zones from server to server
- No need to keep track of what areas do and do not participate in Daylight Savings Time.

However, regardless of how you want to express your datetime values, CIC always uses UTC internally. Whenever a datetime is converted into its TZ equivalent, the underlying value remains unchanged.

Related Topics

[Creating Datetime Values](#)

[Changing Datetime Values](#)

[Getting Datetime Values](#)

[Comparing Datetime Values](#)

Creating Datetime Values

MakeDateTZ

```
MakeDateTZ(<year>, <month>, <day>, TZ)
```

Creates a DateTime value where the parameters passed in are expressed in the TZ timezone.

If there is an error making the date, the default DateTime of "Jan 1, 1970 12:00:00 AM" server local time is created.

MakeDateUTC

```
MakeDateUTC(<year>, <month>, <day>)
```

Makes a datetime value. The parameters passed in are assumed to be in universal coordinated time.

If there is an error making the date, the default DateTime of "Jan 1, 1970 12:00:00 AM" server local time is created.

MakeDateTimeTZ

MakeDateTimeTZ(<year>, <month>, <day>, <hour>, <minute>, <second>, TZ)

Creates a DateTime value where the parameters passed in are expressed in in the TZ timezone.

If there is an error making the date, the default DateTime of "Jan 1, 1970 12:00:00 AM" server local time is created.

MakeDateTimeUTC

MakeDateTimeUTC(<year>, <month>, <day>, <hour>, <minute>, <second>)

Makes a datetime value. The parameters passed in are assumed to be in universal coordinated time

If there is an error making the date, the default DateTime of "Jan 1, 1970 12:00:00 AM" server local time is created.

ToDT

toDT (String)

Creates a datetime from a string. The string will have one of the following formats:

YYYYMMDDHHmmSSZ

YYYYMMDDHHmmSS

YYYYMMDDHHmmZ

YYYYMMDDHHmm

YYYYMMDDZ

YYYYMMDD

where

YYYY = Year

MM = Month

DD = Day

HH = Hour

mm = Minute

SS = Second

Z = Indicates that this is a UTC datetime.

If there is an error making the date, the default DateTime of "Jan 1, 1970 12:00:00 AM" server local time is created.

ToDTTZ

toDTTZ (String, TZ)

Creates a datetime from the string where the string is expressed . The string will have one of the the formats:

YYYYMMDDHHMMSSZ

YYYYMMDDHHMMSS

YYYYMMDDHHMMZ

YYYYMMDDHHMM

YYYYMMDDZ

YYYYMMDD

where

YYYY = Year

MM = Month

DD = Day

HH = Hour

MM = Minute

SS = Second

Z = Indicates that this is a UTC datetime. If you specify Z at the end of the string for toDTTZ, the timezone is not used.

If there is an error making the date, the default DateTime of "Jan 1, 1970 12:00:00 AM" server local time is created.

Changing Datetime Values

The following operators can be used to change an existing datetime value.

AddYears

`AddYears(dt, years)`

Adds the specified number of years to the datetime value (dt). For example, adding 1 year to January 10, 2003 would result in January 10, 2004.

Note: Each year added does not add 365 days to dt. Instead, each year added adds one to the year portion of that variable. This is done to avoid any problems that may occur by adding through a leap year (e.g., January 4, 2004 to January 4, 2005 is 366 days, not 365). If a non-existent date results, such as by adding one year to February 29, 2004, the resulting dt is normalized to the closest matching real date (i.e., February 28, 2005).

AddMonths

`AddMonths(dt, months)`

Adds the specified number of months to the datetime value (dt). Note that this is **not** a fixed number of thirty days, because not all months are thirty days in length. One month added to January 6 would become February 6, and one month added to February 6 would become March 6, even though January is 31 days in length and February is 28. Also, non-existent dates are "normalized" to the closest matching real date; adding one month to January 31 would result in February 28 (or February 29 if it's a leap year), and adding five months to January 31 would result in June 30.

AddDays

`AddDays(dt, days)`

Adds 86,400 seconds times the number specified in "days" to the datetime (dt). For example, `AddDays(dt, 4)` would add 345,600 seconds to dt.

AddHours

`AddHours(dt, hrs)`

Adds 3600 seconds times the number specified in "hrs" to the datetime (dt). For example, `AddHours(dt, 4)` would add 14,400 seconds to dt.

AddMinutes

`AddMinutes(dt, mins)`

Adds 60 seconds times the number specified in "mins" to the datetime (dt). For example, `AddMinutes(dt, 4)` would add 240 seconds to dt.

AddSeconds

`AddSeconds(dt, secs)`

Adds the number of seconds specified in secs to the datetime (dt).

Getting Datetime Values

The following operators can be used to retrieve an existing datetime value.

GetYearTZ

`GetYearTZ(DateTime, TZ)`

Returns the year for the given datetime if the datetime were expressed in the TZ timezone.

If there is an error obtaining the year, -1 is returned.

GetYearUTC

```
GetYearUTC( DateTime )
```

Returns the year for the given datetime if the datetime were expressed in universal coordinated time.

If there is an error getting the year, -1 is returned.

GetMonthTZ

```
GetMonthTZ( DateTime, TZ )
```

Returns the month for the given datetime if the datetime were expressed in the TZ timezone.

If there is an error obtaining the month, -1 is returned.

GetMonthUTC

```
GetMonthUTC( DateTime )
```

Returns the month for the given datetime if the datetime were expressed in universal coordinated time.

If there is an error getting the month, -1 is returned.

GetDayTZ

```
GetDayTZ( DateTime, TZ )
```

Returns the day for the given datetime if the datetime were expressed in the TZ timezone.

If there is an error obtaining the day, -1 is returned.

GetDayUTC

```
GetDayUTC( DateTime )
```

Returns the day for the given datetime if the datetime were expressed in universal coordinated time.

If there is an error getting the day, -1 is returned.

GetDOWTZ

```
GetDOWTZ( DateTime, TZ )
```

Returns the day of week for the given datetime if the datetime were expressed in the TZ timezone.

1=Sunday, 2=Monday, etc. etc.

If there is an error obtaining the day of week, -1 is returned.

GetDOWUTC

```
GetDOWUTC( DateTime )
```

Returns the day of week for the given datetime if the datetime were expressed in universal coordinated time.

1=Sunday, 2=Monday, etc. etc.

If there is an error getting the second, -1 is returned.

GetHourTZ

```
GetHourTZ( DateTime, TZ )
```

Returns the hour for the given datetime if the datetime were expressed in the TZ timezone.

If there is an error obtaining the hour, -1 is returned.

GetHourUTC

```
GetHourUTC( DateTime )
```

Returns the hour for the DateTime as expressed in universal coordinated time.

If there is an error getting the hour, -1 is returned.

GetMinuteTZ

```
GetMinuteTZ( DateTime, TZ )
```

Returns the minute for the given datetime if the datetime were expressed in the TZ timezone.

If there is an error obtaining the minute, -1 is returned.

GetMinuteUTC

```
GetMinuteUTC( DateTime )
```

Returns the minute for the DateTime as expressed in universal coordinated time.

If there is an error getting the minute, -1 is returned.

GetSecondTZ

```
GetSecondTZ( DateTime, TZ)
```

Returns the second for the given datetime if the datetime were expressed in the TZ timezone.

If there is an error obtaining the second, -1 is returned.

GetSecondUTC

```
GetSecondUTC( DateTime )
```

Returns the second for the DateTime as expressed in universal coordinated time.

If there is an error getting the second, -1 is returned.

DateStrTZ

```
DateStrTZ (DateTime, TZ )
```

Returns a fixed length string representing the DateTime expressed in the TZ timezone with the format:

```
YYYYMMDDHHmmSS
```

where

```
YYYY = Year
```

```
MM = Month
```

```
DD = Day
```

```
HH = Hour
```

```
mm = Minute
```

```
SS = Second
```

If an invalid timezone is passed in or there is an error creating the string, the returned datetime string is "".

DateStrUTC

DateStrUTC (DateTime)

Returns a fixed length string representing the DateTime, expressed in universal coordinated time, with the format:

YYYYMMDDHHmmSSZ

where

YYYY = Year

MM = Month

DD = Day

HH = Hour

mm = Minute

SS = Second

Z = Indicates that this is a UTC datetime

If there is an error creating the string, the returned datetime string is "".

Related Topics

[Date/Time Functions - Using Time Zones versus Universal Coordinated time \(UTC\)](#)

Comparing Datetime Values

The date comparison functions for years and months return a value that represents the difference in dates only for the relevant component of the date. For example, in performing #DiffYearsTZ or #DiffYearsUTC, it is the actual year that is being looked at, not the number of 365 day periods that have elapsed between the two dates. So, for example, if #DiffYearsTZ is performed on the dates December 30th, 1997 and January 2nd, 1998, it would return a value of 1 even though the two dates are only three days apart.

The date comparison functions for days, hours, minutes, and seconds, however, convert each date to a numeric value and then return a value that represents the date difference in a number of units of seconds (60 seconds for DiffMinutes, 3,600 seconds for DiffHours, and 86,400 seconds for DiffDays).

DiffYearsTZ

DiffYearsTZ(DateTime1, DateTime2, TZ)

Returns an integer to indicate the year difference between DateTime1 and DateTime2, as expressed in the timezone TZ.

The "year difference" can be thought of as the absolute value of:

GetYearTZ(DateTime1, TZ) - GetYearTZ(DateTime2, TZ)

If there is an error calculating the difference, -1 is returned.

DiffYearsUTC

DiffYearsUTC(DateTime1, DateTime2)

Returns an integer to indicate the year difference between DateTime1 and DateTime2, as expressed in UTC.

The "year difference" can be thought of as the absolute value of:

GetYearUTC(DateTime1) - GetYearUTC(DateTime2)

If there is an error calculating the difference, -1 is returned.

DiffMonthsTZ

DiffMonthsTZ(DateTime1, DateTime2, TZ)

Returns an integer to indicate the month difference between DateTime1 and DateTime2, as expressed in the TZ timezone.

The "month difference" can be thought of as the absolute value of:

GetYearTZ(DateTime1, TZ) * 12

+ GetMonthTZ(DateTime1, TZ) - (GetYearTZ(DateTime2, TZ) * 12

`+GetMonthTZ (DateTime2, TZ)`

If there is an error calculating the difference, -1 is returned.

DiffMonthsUTC

`DiffMonthsUTC (DateTime1, DateTime2)`

Returns an integer to indicate the month difference between DateTime1 and DateTime2, as expressed in UTC.

The "month difference" can be thought of as the absolute value of:

`(GetYearUTC(DateTime1))*12`

`+GetMonthUTC (DateTime1)) - (GetYearUTC (DateTime2) *12`

`+GetMonthUTC (DateTime2))`

If there is an error calculating the difference, -1 is returned.

DiffDays

`DiffDays (DateTime1, DateTime2)`

Returns an integer to indicate the number of 86,400-second units between the two DateTime values. (86,400 = 24 hours x 60 minutes x 60 seconds)

DiffHours

`DiffHours (DateTime1, DateTime2)`

Returns an integer to indicate the number of 3,600-second units between the two DateTime values. (3,600 equals 60 minutes x 60 seconds.)

DiffMinutes

`DiffMinutes (DateTime1, DateTime2)`

Returns an integer to indicate the number of 60-second units between the two DateTime values.

DiffSeconds

`DiffSeconds (DateTime1, DateTime2)`

Returns an integer to indicate the number of seconds between the two DateTime values.

Related Topics

[Date/Time Functions - Using Time Zones versus Universal Coordinated time \(UTC\)](#)

List Operators

GetAt

`GetAt (ANY_LIST_TYPE, Integer)`

Retrieves the value of a specific element in a list of any type. The value retrieved is of the same type as the list. If the position you are retrieving from is empty, or if the element does not exist, this operation will return a null value.

Example: The ListOfString contains "dog" in position zero, "cat" in position one, and "mouse" in position two. `GetAt (ListOfString, 1)` results in a string with a value of "cat".

If there is no value at the specified position, this operation will return one of the following default values:

- List of Integer - 0
- List of Numeric - 0
- List of Boolean - false
- List of String - ""

- List of DateTime - Jan 1, 1970, 0,0,0

Note: There is no way to tell if a GetAt failed or succeeded and the value is just default unless you check the value returned by a GetCount.

GetHead

GetHead (ANY_LIST_TYPE)

Retrieves the value of the first element in a list of any type. The value retrieved is of the same type as the list. If the position you are retrieving from is empty, or if the element does not exist, this operation will return a null value.

Example: The ListOfString contains "dog" in position zero, "cat" in position one, and "mouse" in position two. GetHead (ListOfString) results in a string with a value of "dog".

If there is no value at the specified position, this operation will return one of the following values:

- List of Integer - 0
- List of Numeric - 0
- List of Boolean - false
- List of String - ""
- List of DateTime - Jan 1, 1970, 0,0,0

Note: There is not way to tell if a GetHead failed or succeeded and the value is just 0 unless you check the value returned by a GetCount.

GetTail

GetTail (ANY_LIST_TYPE, ANY_LIST_TYPE)

Retrieves the value of the last element in a list of any type. The value retrieved is of the same type as the list. If the position you are retrieving from is empty, or if the element does not exist, this operation will return a null value.

Example: The ListOfString contains "dog" in position zero, "cat" in position one, and "mouse" in position two. GetTail (ListOfString) results in a string with a value of "mouse".

If there is no value at the specified position, this operation will return one of the following values:

- List of Integer - 0
- List of Numeric - 0
- List of Boolean - false
- List of String - ""
- List of DateTime - Jan 1, 1970, 0,0,0

Note: There is not way to tell if a GetTail failed or succeeded and the value is just 0 unless you check the value returned by a GetCount.

GetCount

GetCount (ANY_LIST_TYPE, Integer)

Counts the number of elements in a list and returns that count as an integer value.

Example: The ListOfString contains the "dog" in position zero, "cat" in position one, and "mouse" in position two. GetCount (ListOfString) results in an Integer with a value of three.

Find

Find (ANY_LIST_TYPE, ANY_SIMPLE_TYPE, Integer)

Looks for a specific value in a list of values of the same type. The position of the first matching value is returned as an integer value. The input integer value is the number of elements in the list to skip before starting the search. To skip the first three elements in a list, you would use the input integer value of three. If the find operation is unsuccessful, this operation returns -1.

Example: The ListOfString contains "dog" in position zero, "cat" in position one, and "mouse" in position two. Find (ListOfString, "Cat",0) results in an output of 1. In another example, the ListOfString contains "apple" in position zero, "orange" in position one,

"banana" in position two, and "apple" in position three. Find (ListofString, "apple", 2) results in an output of 3.

Literal values

Literal values

Expression Editor Assistant provides several literal values you might need as you build expressions. These values are described below:

Boolean values

True	assigns a value of true
False	assigns a value of false

Integer values

0	assigns a value of 0
1	assigns a value of 1
2	assigns a value of 2

Numeric values

0.0	assigns a value of 0.0
1.0	assigns a value of 1.0

String values

" "	space
""	empty string
"\""	quotation mark
"\n"	new line
"\r"	return
"\t"	Tab

Mathematical operators

String Operators

The following operators perform string operations.

? & ?

`(String) & (String)`

Appends the second String onto the first String.

Example: Assume that String1 has a value of "Hokey" and String2 has value of "Pokey". `(String1) & (String2)` results in a String value of "HokeyPokey".

SQLStr()

`SQLStr (String or DateTime or Integer or Numeric or Boolean)`

Converts the input value, which can be any Normal data types (except handle types), into a properly formatted string suitable for direct insertion into a SQL command. The behavior depends on the input data type.

Examples:

String: Doubles any embedded single quotes, then single quote surrounds the result before returning. For example:

Input: Fred's Bar and Grill

Output: 'Fred"s Bar and Grill'

This also eliminates some messy single quote manipulations. For example, the following expression:

```
"WHERE LastName = " & myLastNameVar & ""
```

can be replaced with:

```
"WHERE LastName = " & SQLStr(myLastNameVar)
```

This last expression is also better since it will correctly handle embedded single quotes.

DateTime: Returns an ODBC timestamp escape string of the form:

```
{ts 'YYYY-MM-DD hh:mm:ss'}
```

Using SQLStr ensures portability across RDBMSs (as long as the ODBC driver supports timestamp escape processing – which most do nowadays).

Integer: Returns the string representation of the number.

Numeric: Returns the string representation of the number, using the default precision and scale of the Java `Double.toString()` function. This will work for most numeric types; however, if you need to provide a particular precision and/or scale, then SQLStr cannot be used (i.e. you must perform the conversion manually).

Boolean: Returns 1 if true and 0 if false. This works when the RDBMS type is a numeric or bit type; however, if the type is represented as a character, then SQLStr cannot be used (i.e. you must perform the conversion manually).

StrEqNoCase

`StrEqNoCase (string, string)`

Performs a case insensitive comparison of the two strings.

Example: A condition step with the condition, `StrEqNoCase ("IAmSoCool", "iamsocool")` will take the True exit path.

StrLeft

`StrLeft (String, Integer)`

Returns a specific number of letters from a String value starting from the left.

Example: `StrLeft ("Elephant", 3)` returns a string with a value of "Ele".

StrLen

`StrLen (String)`

Determines the number of characters in a string value.

Example: `StrLen ("Coffee")` results in an integer with a value of six.

StrLower

`StrLower (String)`

Converts the letters of a String value to all lowercase.

Example: `StrLower ("John Doe")` results in a String with a value of "john doe".

StrMid

`StrMid (String, Integer, Integer)`

Returns the characters between two positions.

The *first* integer specifies the starting position. (0 is the first position.)

The *second* integer specified the ending position.

Note: If the string is shorter than the second (ending) position, only the length of the string is returned.

Example: `StrMid ("Coffee", 2, 3)` returns a String with a value of "ff".

`StrMid ("Submarine", 0, 6)` returns a String with a value of "Submari".

`StrMid("coffee", 2, 10)` returns a String with a value of "ffee"

StrPos

`StrPos (String, String, Integer)`

Finds the first occurrence of a String value within a String value starting at a specific position (the first position is zero). If the string is not found, `StrPos` returns -1.

Example: `StrPos ("Coffee", "f", 0)` results in an Integer value of 2. The first "f" occurs at position two in the String "Coffee".

StrRight

`StrRight (String, Integer)`

Returns a specific number of letters from a String value starting from the right.

Example: `StrRight ("Elephant", 3)` returns a String with a value of "ant".

StrTrim

`StrTrim (String)`

Removes any leading and trailing spaces from a String value.

Example: `StrTrim(" 123 456 ")` would result in the value "123 456".

StrUpper

`StrUpper (String)`

Converts the letters of a String value to all upper case.

Example: `StrUpper ("John Doe")` results in a String with a value of "JOHN DOE".

SubstStr

`SubstStr (String1,String2,String3)`

Substitutes String3 for String2 if String2 occurs inside of String1.

String1 is the quoted string or string variable to search.

String2 is the quoted string or string variable to search for in String1.

String3 is the quoted string or string variable to replace String2 in String1.

Example: SubstStr("Welcome to Interactive Marketing","Marketing","Research") returns "Welcome to Interactive Research".

SubstStrRegEx

SubstStr (String1,RegEx,String3)

Substitutes String3 for any matches found by the RegEx (regular expression) inside of String1.

String1 is the quoted string or string variable to search.

RegEx is the regular expression used to search for a sub-string in String1.

String3 is the quoted string or string variable to replace any match found by the RegEx in String1.

Example: SubstStrRegEx("What type of value is this: 12.202.48.80", "\\b\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}\\b", "IP Address") returns a value of "What type of value is this: IP Address".

Note: To get a single slash in the actual string at runtime, you need to enter double slashes. For example, to get the string "+" you need to enter "\\+" in the Interaction Designer editor.

Type conversion functions

Type Conversions

The following operators convert values from one type to another.

toR

`toR (Integer or String)`

Converts an Integer or String value into a Numeric value.

Example: `toR (4)` results in a value of 4.0.

Round

`Round (Numeric)`

Converts a Numeric value to an Integer value by rounding to the nearest whole number.

Example: `Round(4.32)` results in a value of 4.

`Round(4.50)` results in a value of 5.

`Round(4.73)` results in a value of 5.

toS

`toS (Integer or Numeric)`

Converts an Integer or Numeric value into a String value.

Example: `toS (123)` results in a string with a value of "123".

toI

`toI (String or Numeric)`

Converts a String or Numeric value into an Integer value.

Example: `toI ("456")` results in an Integer with a value of 456.

Note: This operation will fail if the string contains information that cannot be converted to an integer. This includes letters or decimal information.

Test (?, ?, ?)

`Test (Boolean, [string or integer or numeric or datetime], [string or integer or numeric or datetime])`

Tests a Boolean condition and returns the value of either the second or third parameter. If the value of the first parameter is true, the value of the second parameter is returned. If the value of the first parameter is false, the value of the third parameter is returned. This is similar to an If...Then statement.

Example: `Test (6>5, "yes", "no")` returns a value of the second parameter or "yes".

`Test (6>7, "yes", "no")` returns a value of the third parameter or "no".

Constants

Constants

Use the following constants when creating expressions in handlers:

\$ComputerName

Returns the name of the computer where the handler is running

\$CurrentStepID

Returns an integer that contains the current step ID.

\$HandlerName

Returns the name of the currently running handler.

\$HandlerStack

Returns a string with the following format:

```
<HandlerName>:<StepID>| [<HandlerName>:<StepID>| ]
```

This is the current step's executing handler stack.

For example, if step 5 in handler A calls handler B and the currently executing step is step 6 in handler B, this constant would return the following string:

```
B: 6 | A: 5 |
```

The order of the handlers listed in the string from left to right goes from the currently executing handler and works its way up the stack.

\$Now

Returns the current time and date.

\$PreviousStepID

Returns an integer that contains the step ID for the step that lead in to the currently active step.

Miscellaneous topics

Print Preview toolbar

The print preview toolbar offers you the following options:

Print

Bring up the print dialog box, to start a print job.

Next Page

Preview the next printed page.

Prev Page

Preview the previous printed page.

One Page / Two Page

Preview one or two printed pages at a time.

Zoom In

Take a closer look at the printed page.

Zoom Out

Take a larger look at the printed page.

Close

Return from print preview to the editing window.

Status Bar



The status bar is displayed at the bottom of the Interaction Designer window. To display or hide the status bar, use the Status Bar command in the View menu.

The left area of the status bar describes actions of menu items as you use the arrow keys to navigate through menus. This area similarly shows messages that describe the actions of toolbar buttons as you depress them, before releasing them. If after viewing the description of the toolbar button command you want not to execute the command, then release the mouse button while the pointer is off the toolbar button.

The area on the right side of the status bar indicates which of the following keys are on:

Indicator	Description
CAP	Caps Lock is on.
NUM	Num Lock is on.
SCRL	Scroll Lock is on.

Compile Voicemail TUI Initiator

Email Tool Result Codes

The Result output from the Email tools is an integer that indicates either success or error, as well as the result type.

This integer is for advanced diagnostic purposes and is to be used by PureConnect Customer Care.

Integer	Success/Error	Result Description
0	Success	Result
1	Error	Unknown result
2	Success	Found Result
3	Success	Found maximum result
4	Success	Not found result
5	Success	Canceled result
6	Success	Pending result
7	Success	Valid action
8	Success	Invalid action
9	Error	Private result
10	Error	Timeout result
11	Error	Invalid message file result

12	Error	Invalid sender result
13	Error	Missing recipient result
14	Error	Invalid path result
15	Error	Invalid attachment result
16	Error	Invalid index result
17	Error	Not found result
18	Error	Server down result
19	Error	Folder closed result
20	Error	Invalid mailbox result
21	Error	Invalid folder result
22	Error	Invalid cookie result
23	Error	Invalid status result
24	Error	Notifier FTP result
25	Error	Failed HRESULT result
26	Error	Notifier exception result
27	Error	CException result
28	Error	SEH result
29	Error	Win32 exception result
30	Error	COM error result
31	Error	Exception result
32	Error	Unknown exception result
33	Error	Not supported result
34	Error	Valid unsent recipient result
35	Error	No transports result
36	Error	Invalid interaction result
37	Error	Invalid moniker result
38	Error	Invalid URI result
39	Error	Invalid host result
40	Error	Compression timeout result
41	Error	Access denied result
42	Error	Has messages result
43	Error	Has folders result
44	Error	Invalid message

45	Error	Search limit exceeded
----	-------	-----------------------

Align Bottom

Aligns all selected toolsteps so that the bottom margin of each toolstep is in the same line with the bottom margin of the primary selected tool.

Align Left

Aligns all selected toolsteps so that the left margin of each toolstep is in the same line with the left margin of the primary selected tool.

Align Right

Aligns all selected toolsteps so that the right margin of each toolstep is in the same line with the right margin of the primary selected tool.

Align Top

Aligns all selected toolsteps so that the top margin of each toolstep is in the same line with the top margin of the primary selected tool.

Exit command (File menu)

Use this command to end your Interaction Designer session. You can also use the Close command on the application Control menu. Interaction Designer prompts you to save documents with unsaved changes.

Shortcuts

Mouse: Double-click the application's Control menu button.

Keys: ALT+F4

Associate Call

Associates an incoming call with a campaign list entry and returns the detailed data for the caller from the Interaction Dialer Campaign COM object.

Caution: Do not use this tool in your handlers as it may cause CIC to function incorrectly. It is intended for Interaction Director handlers only. You will never need to use or modify the values set in this tool.

Inputs

Phone Number

Key used to look up this person in the campaign database. The phone number is the unique key that identifies this caller and will be used to look up their detailed data from the database

Call ID

Current Call Id.

Outputs

Campaign ID

The name of the current campaign that this call is associated with.

Request ID

Internal Dialer Id representing the campaign request that contains the current call. (A request can have multiple calls associated with it).

Reference ID

Internal Dialer Id representing the call itself.

Workgroup

ACD workgroup associated with the campaign.

Line Group

Linegroup associated with the campaign.

Counter

Additional field provided for the end user. Used to capture any counter type activity they require on a call by call basis.

Revenue

Additional field provided for the end user. Used to capture any revenue-related activity with the call.

Attempt

Number of times this person has been called while the campaign has been running

Predictive Attribute NameList

List of additional attribute names the user has created by extending the campaign call list table

Predictive Attribute Value List

List of additional attribute values the user has created by extending the campaign call list table.

Exit Paths

Success

This path is taken if the association is successfully made and the specified data returned.

Failure

This path is taken if the operation fails.

Auto Label Power Tools

Two Auto Label power tools are available in Interaction Designer for advanced users.

After you enable the auto label tools, if you edit one of the tools on the tool palette, the label on the tool changes to the contents of the step. Existing steps do not change, but all newly edited step labels change in any handler. You may have to expand the step to see the entire label.

- The **Auto Label Assignments** tool affects the assignment tool steps in a handler.
- The **Auto Label Conditions** tool affects the condition tool steps in a handler.

To enable one of these tools:

- From the **Utilities** menu, point to **Power Tools**, and then choose either the **Auto Label Assignments** or **Auto Label Conditions** tool.

See Also

[Relabel Power Tools](#)

BMP file format for faxes

The [Create Fax Page List](#) and [Append Page tools](#) supports the following BMP file formats:

- Windows Bitmap (BMP). This is a file format created by Microsoft. Some BMP images are compressed with an RLE-type compression.

You can read the following bits per pixel, without RLE compression: 1, 4, 8, 16, 24, 32.

You can read the following bits per pixel, with RLE compression: 1, 4, 8.

- OS/2 Bitmap (OS/2 BMP). These are files created on an OS/2 operating system. LEADTOOLS supports both 1.x and 2.x formats.

You can read the following bits per pixel, without RLE compression: 1, 4, 8, 24.

You can read the following bits per pixel, with RLE compression: 1, 4, 8.

Call ID to Integer

This tool is no longer part of the tool palette. It was replaced in version 2.4 by the [Convert Call Id to String](#) tool.

Call Info Request

This SMDI tool was deprecated in 4.0.

Callback

A callback is a request sent from a customer via a web site. The request contains contact information for the customer and optionally information pertaining to the nature of the request. When the request is received, it is routed to an agent and a call is made from that agent's station to the sender of the callback request.

Calls to Subroutines

This step is a call to a subroutine. Subroutines are simply handlers started by other handlers, as opposed to handlers started by events. When a handler author publishes a handler started by the Subroutine initiator, a tool to call that subroutine appears on the subroutine page of the [design palette](#). This "subroutine author" determines what the inputs and outputs for the resulting subroutine tool will be by defining the subroutine parameters in the Subroutine initiator. The author flags each parameter as either "input only" or "input/output." The subroutine author can also determine default values for those parameters.

If you don't see an icon for this step, it means the called subroutine is not registered on the server against which you are running Interaction Designer.

Note: If you open a handler containing a call to a subroutine, and the parameters required by the called subroutine have changed, you'll receive a Warning message telling you which parameters have changed.

Inputs

Any values in the input parameters listed here are passed to the called subroutine. The subroutine cannot change these values because they are *input only* values.

Outputs

Any values in the Output parameters listed here are passed to the called subroutine. The subroutine can change these values because they are *input/output* values. The values of any output parameters could change after the called subroutine executes. It's not always necessary to place a value in an output parameter because the called subroutine provides the value upon execution.

Exit Paths

Calls to subroutines always exit along the Next exit path.

Related Topics

[Introduction to Subroutines](#)

[Add or edit a subroutine parameter](#)

Call Stack

This window contains the call stack for the currently executing handler. The information in the call stack is in the format "(step ID) Handler name". Users can select entries in the window to view other handlers that are in the call stack. The calling step is highlighted with a dashed line. The currently executing step will have a solid line around it.

Related Topics

[Debug Palette](#)

[Handler Variables](#)

[Step Variables](#)

[View the value of a variable in a debug handler](#)

[Watch Variables](#)

Choosing an Operator

The box on the left contains a list of operator categories to select from when building your expression. If you select the Variables category, you see a list of variables of the expected type in the Symbols and Values list. If you choose the Literal Values category, you'll see a list of literal values to choose from. Selecting Mathematical operations displays a list of mathematical operations to choose from.

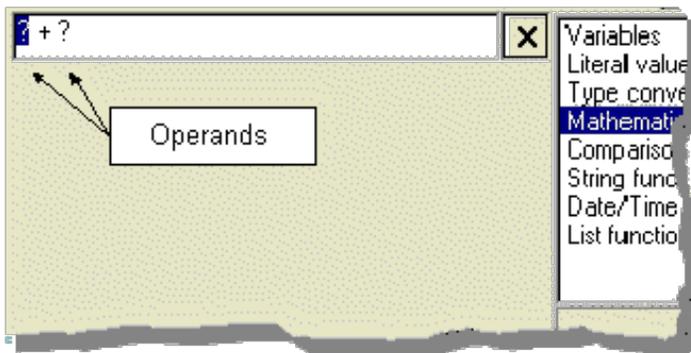
The Expression Editor Assistant limits your choice in variables and operations to those that are of a type valid for the parameter. For example, if the parameter accepts only a Boolean variable (true or false), then the only variables that appear in the Variables list are variables of type Boolean. The only expressions you can build are those that result in a Boolean value.

The first category to choose from is Variables. You can assign a variable as the value of a parameter. To select a variable as the value of a parameter, click on the Variable category. A list of variables appears on the choices box. Double-click on the variable you want to assign as the value of the parameter. That variable appears in the Expression field.

Typing values for operands

Once you've selected a category that contains operations, you can select an operator from the list of choices that appear. For the Mathematical Operators, your choices are "-", "*", "/", "^", "+", "Abs", and "Mod." Double-clicking on an operator places that operator in the expression field. The "?" symbol that appears on either side of the operator are operands. These are placeholders for values that complete the expression. The following figure shows two operands for the "+" operator. It is in these operands that you will place the values for the expression.

You can enter values or build sub-expressions for operands just as you would for a parameter. The only limit is that the end result of the entire expression must be of a type valid for the parameter.



Related Topics

[Introduction to Expression Builder](#)

[Operator Descriptions](#)

Conference ID to Integer

This tool is no longer in the tool palette. It was replaced in version 2.4 by the [Convert Conference ID to String](#) tool.

Context Help command



Use the Context Help command to obtain help on some portion of Interaction Designer. When you choose the Toolbar's Context Help button, the mouse pointer changes to an arrow and question mark. Click somewhere in the Interaction Designer window, such as another Toolbar button, to display the Help topic for that item or command.

Shortcut

Keys: SHIFT+F1

Debug Palette

The debug palette contains four tabs for viewing the activity of a handler during the debug session. The [Watch Variables](#), [Step Variables](#), and [Handler Variables](#) tabs show the values of variables throughout the debugging session. The [Call Stack](#) tab shows the call stack for the debug session.

Related Topics

[View the value of a variable in a debug handler](#)

Debug Preferences

Use this tab of the Designer Preferences dialog box to set Interaction Designer's default behavior during debugging sessions.

To display the Designer Preferences dialog box, click the Edit menu, click Preferences, and then click Designer Preferences.

Initial Breakpoint

This setting determines where the initial breakpoint will be in handlers where the initiator's exit path leads directly to a Notify Debugger step. The three options are:

- **Initiator** – The handler will automatically pause on the initiator when the debugging session begins.
- **Notify Debugger** – The handler will first pause on the Notify Debugger step when the debugging session begins. This means that the initiator will have already executed before the first breakpoint.
- **Prompt** – The user will be prompted to set the initial breakpoint at the beginning of each debugging session.

Handlers that do not have a Notify Debugger step immediately following the initiator will always use the initiator as the initial breakpoint.

When Stopping a Debug Session

This determines ID's behavior when a debugging session is stopped before the handler has completed. The three options are:

- **Stop Handler Execution** – IP will perform no further processing on the handler.
- **Continue Handler Execution** – IP will take over processing of the handler and run it to its normal completion.
- **Prompt** – The user will be prompted to choose one of the above two options.

Note: if the handler is stopped by closing the debug window, the handler will continue in IP regardless of these settings.

Watch Variables

Users may select how much information they want to see on their watched variables.

- **Display hex value for integers** – If checked, the hex value of any integers will be shown in addition to their decimal values.
- **Display string length for strings** – If checked, the length of string variables will be shown in addition to their values.

Related Topics

[Designer Preferences Page](#)

[General Preferences](#)

[Handler Download Preferences](#)

[Views Preferences](#)

Debug Toolbar

Click	To
	Causes a handler to run until it reaches the next break point or until the session ends.
	Stops the debug session. When the debugging session is stopped by this method, the user will be prompted if they want to allow the handler to continue when the debugging session ends.
	Moves to the next step in the handler. This button is only used when single-stepping through handlers.
	Press this button to enter a subroutine in the debugging session. This button is only available when the handler is paused on a subroutine step.
	Press this button to step out of a subroutine.
	Press this button to jump to the currently executing step in the handler.
	<p>Allows you to add a handler to the debugging session. Click the button to add handlers to the debugging session, and use the drop-down to view handlers currently loaded. Selecting a handler from the drop-down menu will bring that handler to the foreground. Handlers that are listed with a "Level #" next to them are in the current call stack. Handlers with no "Level #" are loaded in the debug session window, but are not in the current call stack.</p> <p>Handlers are automatically downloaded by Designer and placed in the debug session as they are executed. Adding handlers manually can be helpful if users want to add a handler to a debug session and set a breakpoint on a step in that handler in advance.</p>
	Restarts the debugging session with all the same settings (breakpoints, viewed variables, etc.).

Call Wrap Up

This initiator launches whenever a call is completed by an agent or by the system.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Predictive Call

Object ID

Select {all} for this initiator to work with all Predictive Call objects.

Notification Event

Select Dialer Call Wrap Up.

Outputs

Workflow Name

String containing the name of the workflow that this fax request is associated with.

Campaign Name

String containing the name of the campaign that this fax request is associated with.

PredictiveAttributeNameList

List of attribute names that are associated with this contact and are sourced from the call list table in the Dialer database.

PredictiveAttributeValueList

List of attribute values that are associated with this contact and are sourced from the call list table in the Dialer database.

Exit Paths

Start

This step always takes the Start exit path.

Fax Message Initiator

This initiator is used by the dialer for placing all associated fax calls. A fax call can be generated from either an "agentless" campaign or a regular campaign where fax call analysis returns are associated with a specific .i3f fax file.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Predictive Call

Object ID

Select {all} for this initiator to work with all Predictive Call objects.

Notification Event

Select Dialer Fax Message Request.

Outputs

Workflow Name

String containing the name of the workflow that this fax request is associated with.

Campaign Name

String containing the name of the campaign that this fax request is associated with.

Row Id

String containing the unique identifier for this contact. The Row Id is the identifier that uniquely distinguishes this contact from others in the Call List.

Call ID

The unique identifier for the interaction.

Fax Message File

The .i3f file that will be sent to the answering fax machine.

Number Retries

The number of retries to attempt in the event of an error in transmission.

Exit Paths

Start

This step always takes the Start exit path

Get Logged In Workflow

This Dialer tool returns the campaign and workflow that a specified logged in agent is associated with.

Note: This tool only appears in the tool palette if Dialer is installed.

Inputs

Agent ID

The ID of the logged in agent.

Outputs

Workflow Name

The name of the workflow the agent is associated with.

Campaign Name

The name of the campaign the agent is associated with.

Exit Paths

Next

This tool always takes the Next exit path.

Get Workflows

This Dialer tool returns a list of active workflows defined by Interaction Dialer.

Note: This tool only appears in the tool palette if Dialer is installed.

Outputs

Workflow UUID List

List of UUIDs of all active workflows.

Workflow Name List

List of strings containing the names of all active workflows.

Campaign Name List

List of strings containing the names of all active campaigns.

Exit Paths

Next

This tool always takes the Next exit path.

Rule Action Event

This initiator is used by Dialer to complete specific rule actions or policy behaviors. For example, if a rule action or policy behavior specifies that an email be generated then this handler will be called to perform the physical action of sending the email using the [email](#) tools. Specific actions that can be performed by handlers with this initiator include sending an email or page, or initiating a custom handler.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Predictive Call

Object ID

Select {all} for this initiator to work with all Predictive Call objects.

Notification Event

Select Dialer Rule Action Event.

Outputs

Workflow Name

String containing the name of the workflow that this fax request is associated with.

Campaign Name

String containing the name of the campaign that this fax request is associated with.

Dialer Rule Name

The name of the Rule or Policy that initiated the handler.

Dialer Action Type

Integer indicating the type of action that will be performed. Valid values and the actions attributed to them are described below:

Value	Action Type
1	Send an email.
2	Send a page.
3	Custom handler from a Rule Action.
4	Custom handler from a Pre-call Policy Behavior.
5	Custom handler from a Call Analysis Policy Behavior.
6	Custom Handler from a Disposition Policy Behavior.

Address

The address that is being used by the rule action or policy.

Subject

The subject of the message that is being sent by the rule action or policy.

Message

The textual message of the rule action or policy (e.g. email body)

Evaluation Summary

Internal use only

Exit Paths

Start

This step always takes the Start exit path.

Disconnect Email Object

This tool was deprecated in CIC 4.0.

External Document Not Found

The document you have requested was not found in the IC_Documents directory on your IC Server. This can occur for several reasons:

- you are running this help file from a computer not connected to the IC Server. The IC_Documents is located on your IC Server. If you want the external documents links to work correctly, copy this directory to the same folder from which you are running this help file.
- the application for viewing this file is not installed. You must install the viewing application associated with the file. Acrobat Reader is required for viewing PDF files and is available at www.adobe.com.

Drag a tool from the tool palette to create a step

A step is an instance of a tool in a handler or subroutine. You create steps by dragging tools from the tool palette into a handler window.

To create a step:

1. On the Tools page of the Design palette, click the tab that contains the tool you want to use.
2. Click the tool and drag it to the handler or subroutine you are building.

The step appears in the handler. You can move this step to another location in the handler by clicking the step and dragging it to its new location.

Introduction to E-mail Objects

The Email Object tools were deprecated in 4.0. Additional tools were added to the Email tool palette. For more information, see [Email Tools](#).

End TDD

This Telephony tool was deprecated in 4.0.

EPS file formats for faxing

The [Create Fax Page List](#) and [Append Page tools](#) supports the following EPS file formats:

- EPS: PostScript Raster (Encapsulated PostScript). These files are used primarily on PostScript printers. These printers usually offer more variety of fonts and higher resolution than standard laser printers. EPS files will work on any PostScript compatible printer and any end-user application that supports placement of EPS files in its work space.

The image that you read from an EPS file can be either a PostScript raster image or an embedded TIFF image. The image that you write to an EPS file is always a PostScript raster image.

For a PostScript raster image, you can read the following bits per pixel: 1, 8.

For an embedded TIFF image, you can read the following bits per pixel: 1, 4, 8, 16, 24, 32.

Exporting Dependencies to XML

Dependencies shown in the [View Dependencies](#) window may be exported into XML format. The XML output for each dependency listed in the dependency dropdown listbox will be grouped according to the currently selected Group By radio button on the dialog. The XML that is created will be formatted such that it will be readable in a standard text editor, although loading the XML file into a browser for reading may be preferable.

Change Initiator command (File menu)

Use this command to place an initiator into a handler or to replace the current initiator with a new initiator. Select the initiator you want to use from the menu and click OK. If the handler already contained an initiator, it will be removed and replaced with the selected initiator.

Close command (File menu)

Use this command to close all windows containing the active document. Interaction Designer suggests that you save changes to your document before you close it. If you close a document without saving, you lose all changes made since the last time you saved it. Before closing an untitled document, Interaction Designer displays the Save As dialog box and suggests that you name and save the document.

File Debug Immediate

Initiates a [debugging](#) session with the published version of the handler in the active window.

File Download A Handler From Server Command

Use this command to download one or more handlers from the server. By default, Designer will prompt for a destination directory where the handler should be saved. You can change this behavior on the [Designer Preferences](#) dialog.

File Generate i3pub File

Generates an .i3pub file for [intermediate publish](#).

1, 2, 3, 4 command (File menu)

Use the numbers and filenames listed at the bottom of the File menu to open the last four documents you closed. Choose the number that corresponds with the document you want to open.

New command (File menu)

Use this command to create a new document in Interaction Designer.

You can open an existing document with the [Open command](#).

Shortcuts

Toolbar: 

Keys: CTRL+N

Open command (File menu)

Use this command to open an existing document in a new window. You can open multiple documents at once. Use the Window menu to switch among the multiple open documents. See Window 1, 2, ... command.

You can create new documents with the [New command](#).

Shortcuts

Toolbar: 

Keys: CTRL+O

Print Preview command (File menu)

Use this command to display the active document as it would appear when printed. When you choose this command, the main window will be replaced with a print preview window in which one or two pages will be displayed in their printed format. The [print preview toolbar](#) offers you options to view either one or two pages at a time; move back and forth through the document; zoom in and out of pages; and initiate a print job.

Print Setup command (File menu)

Use this command to select a printer and a printer connection. This command presents a Print Setup dialog box, where you specify the printer and its connection.

File Print command

Use this command to print a document. This command presents a Print dialog box, where you may specify the range of pages to be printed, the number of copies, the destination printer, and other printer setup options.

Shortcuts



Toolbar:

Keys: CTRL+P

File Properties

Opens the properties window for the active handler.

Save As command (File menu)

Use this command to save and name the active document. Interaction Designer displays the Save As dialog box so you can name your document.

To save a document with its existing name and directory, use the [Save command](#).

Save command (File menu)

Use this command to save the active document to its current name and directory. When you save a document for the first time, Interaction Designer displays the Save As dialog box so you can name your document. If you want to change the name and directory of an existing document before you save it, choose the [Save As command](#).

Shortcuts



Toolbar:

Keys: CTRL+S

File Export To XML

Generates a text file of the current handler in XML format.

Find

Opens the Search dialog box to help you search a handler for a specific node.

Get Best Site Queue

This tool returns a Site Name and a Queue at that site that can most quickly take the call. Enterprise queues are defined in Interaction Director.

Inputs

Enterprise Queue

The Enterprise Queue to which you want to route the call. Enterprise Queues are made up of one or more Site Queues. For example, you could have an Enterprise Queue of Support that is comprised of (Site-Indy, Queue-Support), (Site-Boca, Queue-Support), (Site-France, Queue-Support). This tool examines the Support queue at each site.

Skills Required

A list of skills required for the call. These skills should be typed as they appear in Interaction Administrator. Skill Names must be defined the same for all sites. You should not define Windows NT at one site and Win NT at another.

Minimum Skill Levels

A list parallel to Skills Required specifying the minimum skill level required for this call. Separate multiple integer values with a semicolon.

Maximum Skill Levels

A list parallel to Skills Required specifying the maximum skill level required for this call. Separate multiple integer values with a semicolon.

Outputs

Site Name

The name of the site that can most quickly answer this call.

Queue Name

The name of the queue at the best site that can most quickly answer this call.

Queue State

The availability of the queue. It will be one of the following values:

1	Available
2	Closed
3	Inactive
4	Connection Lost

Exit Paths

Success

This tool takes the Success exit path if it was able to return a Site Queue.

Unknown Enterprise Queue

This tool takes the Unknown Enterprise Queue exit path if the Enterprise Queue was not valid. Enterprise queues are defined in Interaction Director.

Failure

This tool takes the Failure exit path if it was unable to find a suitable Site Queue. This can occur if no agents have the required skills, the other site queues are closed, there are no available lines, etc.

Get Campaign UUID

Note: This tool is no longer in the tool palette. It was removed in version 2.3.1.

This Dialer tool returns the UUID associated with a campaign.

Caution: Do not use this tool in your handlers as it may cause CC to function incorrectly. It is intended for Interaction Director handlers only. You will never need to use or modify the values set in this tool.

Inputs

Campaign ID

The name of the campaign to query.

Outputs

Predictive Campaign UUID

The UUID associated with the campaign.

Exit Paths

Success

This path is taken if the specified UUID is successfully returned.

Failure

This path is taken if the operation fails.

Get Key Word Set Guid List

This Telephony tool returns the GUIDs for the specified keyword sets.

Note: Before you use this tool, you should check the keyword spotting feature in the Interaction Recorder Policy Editor. As of CIC 4.0 SU3, Policy Editor offers keyword spotting by workgroups, roles, individual users, lines, and stations. Use this tool in situations where keyword spotting in Policy Editor doesn't meet your needs.

Inputs

List of Keyword Set Names

A list of key word set names for which the user wants the associated GUIDs.

Outputs

List of Keyword Sets GUIDs Found

A list of the keyword set GUIDs found for the keyword set names specified as input.

List of Keyword Sets Names Found

A list of the keyword set names found for each of the GUIDs that was found. This list should have the same number of entries as the associated output List of Keyword Sets GUIDs Found.

List of Keyword Sets Names Found

A list of the keyword set names that were not found. If a specified keyword set name doesn't exist, it appears in this list.

Exit Paths

Success

This tool takes the Success exit path if GUIDs were found for all the keyword spotting sets.

Failure

This tool takes the Failure exit path if GUIDs can't be found for all of the keyword sets. In the event of a failure, the tool still goes through the rest of the keyword set names and tries to map them to the GUIDs and still creates an output lists.

Get Site Queue Expected Wait

This tool returns the estimated time before a call can be answered at a Site Queue. Use this tool if you want to perform custom routing and not allow Interaction Director to pick the best site with the [Get Best Site Queue tool](#). A Site Queue is one of the sites that makes up an Enterprise Queue. For example, you could have an Enterprise Queue of Support that is comprised of the following site queues: (Site-Indy, Queue-Support), (Site-Boca, Queue-Support), (Site-France, Queue-Support).

Inputs

Site Name

The site name as listed in Interaction Administrator's Interaction Director Sites container.

Queue Name

The name of the queue as listed in Interaction Administrator's Interaction Director Queues container.

Skills Required

A list of skills required for the call. These skills should be typed as they appear in Interaction Administrator. Skill Names must be defined the same for all sites. You should not define Windows NT at one site and Win NT at another.

Minimum Skill Levels

A list parallel to Skills Required specifying the minimum skill level required for this call. Separate multiple integer values with a semicolon.

Maximum Skill Levels

A list parallel to Skills Required specifying the maximum skill level required for this call. Separate multiple integer values with a semicolon.

Outputs

Estimated Wait Time

An estimate of the amount of time (in seconds) that a call will wait before being connected to an agent. This is calculated for the next call that enters the queue. This wait time is calculated by multiplying the average wait time for the queue times this call's position and then subtracting the amount of time that the call has been on the queue. Average wait time is calculated by adding the total wait time for all calls answered in a given period and dividing that total by the number of calls answered in that period. A negative value is returned if the average wait time statistics are not available (due to insufficient observations in the sampling period) or if the call has already waited longer than the estimated wait time.

Queue State

The availability of the queue. It will be one of the following values:

1	Available
2	Closed
3	Inactive
4	Connection Lost

Exit Paths

Success

This tool takes the Success exit path if it returned the estimated wait time.

Unknown Site

This tool takes the Unknown Site exit path if the Site Name you specified was invalid.

Unknown Queue

This tool takes the Unknown Queue exit path if the Queue Name you specified was invalid.

Failure

This tool takes the Failure exit path if it could not return the estimated wait time. Other reasons this tool might fail include if no agents have the required skills, the other site queues are closed, there are no available lines, etc.

Get Site Queue Info

This tool returns statistical information about a Site Queue. A Site Queue is one of the sites that makes up an Enterprise Queue. For example, you could have an Enterprise Queue of Support that is comprised of the following site queues: (Site-Indy, Queue-Support), (Site-Boca, Queue-Support), (Site-France, Queue-Support).

Inputs

Site Name

The site name as listed in Interaction Administrator's Interaction Director Sites container.

Queue Name

The name of the queue as listed in Interaction Administrator's Interaction Director Queues container.

Skills Required

A list of skills required for the call. These skills should be typed as they appear in Interaction Administrator. Separate multiple skills with a semicolon.

Minimum Skill Levels

A list parallel to Skills Required specifying the minimum skill level required for this call. Separate multiple integer values with a semicolon.

Maximum Skill Levels

A list parallel to Skills Required specifying the maximum skill level required for this call. Separate multiple integer values with a semicolon.

Outputs

Longest Available Agent

The number of seconds that one of the agents monitoring this queue has been available.

Longest Wait

The longest wait of the currently waiting calls.

Average Wait

The average wait time of all calls waiting on this queue.

Average Abandoned

The average length of abandoned calls.

Average Call Length

The average length of all calls on this queue.

Average ACW

The average time agents have been in a follow up status for all the agents currently in follow up status.

Average Period

The number of seconds over which the averaged values are calculated.

Average Wait Sample

The number of calls used to calculate the Average Wait statistic.

Average Abandon Sample

The number of calls used to calculate the abandoned calls statistic.

Average Length Sample

The number of calls used to calculate the Average Call Length statistic.

ACW Sample

The number of agents used to calculate the Average ACW statistic.

Active Lines

The number of active lines on which calls come into this queue.

Free Lines

Of the active lines, the number of lines that do not have calls.

Logged in Agents

The number of agents logged into this queue.

Available Agents

The number of agents with an Available status.

Assigned Agents

The number of agents that are currently connected to calls.

ACW Agents

The number of agents in a follow-up status.

Logged In Agents Matching Skills

The number of agents currently logged in that have skills within the skill parameters you specified.

Calls Waiting

The number of calls waiting on this queue.

Calls Connected

The number of calls connected on this queue.

Service Level

The current service level for this queue. Service level is a percentage calculated by figuring the number of calls answered within a time specified in Interaction Administrator.

Queue State

The availability of the queue. It will be one of the following values:

1	Available
2	Closed
3	Inactive
4	Connection Lost

Exit Paths

Success

This tool takes the Success exit path if it returned the statistics.

Unknown Site

This tool takes the Unknown Site exit path if the Site Name you specified was invalid.

Unknown Queue

This tool takes the Unknown Queue exit path if the Queue Name you specified was invalid.

Failure

This tool takes the Failure exit path if it could not return the statistics.

Get Site Queue State

This tool returns the Queue State for a Site Queue you specify. This tool returns statistical information about a Site Queue. A Site Queue is one of the sites that makes up an Enterprise Queue. For example, you could have an Enterprise Queue of Support that is comprised of the following site queues: (Site-Indy, Queue-Support), (Site-Boca, Queue-Support), (Site-France, Queue-Support).

Inputs

Site Name

The name of the site you want to query.

Queue Name

The name of the queue you want to query at the specified site.

Outputs

Queue State

The availability of the queue. It will be one of the following values:

1	Available
2	Closed
3	Inactive
4	Connection Lost

Exit Paths

Success

This tool takes the Success exit path if the Queue State was returned.

Unknown Site

This tool takes the Unknown Site exit path if the Site Name you specified was invalid.

Unknown Queue

This tool takes the Unknown Queue exit path if the Queue Name you specified was invalid.

Failure

This tool takes the Failure exit path if the queue state was not returned.

Get Email Object Original Message

This tool was deprecated in CIC 4.0.

Get Email Object Response Message

This tool was deprecated in CIC 4.0.

Go TopLeft

Shifts the view in the design window all the way to the top and left.

Go Line Down

Scrolls the design window down by one line.

Go Line Left

Scrolls the design window left by one line.

Go Line Right

Scrolls the design window right by one line.

Go Line UP

Scrolls the design window upward by one line.

Go Page Down

Shifts the view in the design window down by the length of one full screen.

Go Page Left

Shifts the view in the design window left by the width of one full screen.

Go Page Right

Shifts the view in the design window right by the width of one full screen.

Go Page Up

Shifts the view in the design window up by the length of one full screen.

Grid

Toggles the background grid on and off in the design window.

Handler Diff Power Tool

The Handler Diff tool lets you compare two directories of handlers. The output is an XML file that lists the differences between the two sets of handlers. Optionally, you can specify an XSL file that will customize the display of the resulting diff XML file into a more readable form.

To use the tool to compare two directories of handlers:

1. Close all open handlers in Interaction Designer.
2. From the **Utilities** menu, point to **Power Tools**, and then choose **Handler Diff**.

The Handler Diff dialog box appears.

3. Under **"Old" Handler List**, click the **Browse** button and then specify the folder that contains the set of the original handlers you want to use.

After you select the folder, the Handler Diff tool populates the Old list with the .ihd files found in that folder.

4. Under **"New" Handler List**, click the **Browse** button and then specify the folder that contains the set of the original handlers you want to compare with the Old Handler List.
5. Uncheck any handlers that you do not want to process.

Each list also has a button at the bottom of the list (**Check only handlers that are checked in "New" list** and **Check only handlers that are checked in "Old" list**) that will automatically check only those handlers that exist in the other list.

6. On the **Diffs to Perform** tab in the lower portion of the dialog box, clear the check box for any diffs you do not want to perform. This tab lists all the handler attributes that you can compare.

When you click on a diff, a description of that diff appears in the **Diff description** portion of the dialog. There is a **Check All Diffs** button that you can click to select all diffs at once.

7. To change the location of the output diff XML file, click the **Output Location** tab, and then click **Pick output XML file** and specify where you want to have the diff output XML file written.
8. To customize the output with an XSL file, click the **Output Formatting** tab, select the **Add XSL stylesheet to customize handler diff XML output** checkbox, and then select the appropriate .XSL file from the list.

GeneralHandlerDiffOverview.xsl is a sample, non-localized file installed in \IC\Server\HandlerDiff\XSLFiles that gives an example of how you can use an XSL file to customize the output. If you decide to use an XSL file to customize the output, note that the location of the stylesheet is written directly into the diff results XML file with a "<?xml-stylesheet" processing instruction at the top. If you give the output XML file to another user with a stylesheet specified, it may not display the same way on their machine unless the XSL file exists in the location specified in the XML file.

9. Click **Run Handler Diff** to start the comparison.

When the diff operation completes, the diff tool will launch Internet Explorer to display the resulting XML file. If you specified an XSL sheet to use to customize the output, when Internet Explorer opens the file, you might see the error "To help protect your security, Internet Explorer has restricted this file from showing active content that could access your computer. Click here for options..." If this message appears, click the message and then click **Allow Blocked Content** to display the information.

10. Click on a handler name in the left column to display the diff results for that handler in the right column.

Handler Download Preferences

These settings affect the way Interaction Designer downloads handlers by means of the **Download Handler from Server** command on the **File** menu. The settings also affect how handlers are loaded into a debugging session.

Handler Retrieval

This setting indicates the maximum number of seconds that Interaction Designer waits to download a handler. If the handler cannot be loaded in the specified time, the user receives the "Could not retrieve handler from server archive" message box and the retrieval attempt stops.

Destination Directory

This is the directory to which the handler is downloaded. If left unspecified, users are prompted to supply a directory each time a handler is downloaded.

Warn if Handler Already Exists in Destination Directory

If this box is left checked, the user is prompted about whether or not to continue if the handler being downloaded already exists in the specified directory. If this check box is cleared, Interaction Designer overwrites the handler already in the specified directory without prompting the user.

Related Topics

[Debugging Preferences](#)

[Designer Preferences Page](#)

[General Preferences](#)

[Views Preferences](#)

Handler Variables

When execution is paused on a step, users can click on the Handler Variables tab in the Debug Palette to view the variable values for all variables in the currently displayed handler.

Related Topics

[Call Stack](#)

[Debug Palette](#)

[Step Variables](#)

[View the value of a variable in a debug handler](#)

[Watch Variables](#)

How Paging Works in CIC

Paging Services uses the carrier telephone number to select the appropriate carrier out of a list of known paging carriers. The list of carriers is configurable from Interaction Administrator and is maintained by Directory Services. The list is read at initialization and monitored by Paging Services for changes during operation. If a carrier is found in the list, Paging Services uses stored parameters to establish the connection with that carrier. Failure to find a matching carrier will cause the system to attempt to establish a connection anyway using common defaults in the hope that the page can be sent. An event is logged in this case even if the page is sent successfully to indicate that a new carrier should be added to the list in Directory Services.

To most efficiently use telephone lines, non-urgent pages are processed in batches by collecting page requests for a configurable interval, sorting them according to carrier, and then sending as many as possible during a single call to the carrier. There may be carrier-imposed limits as to how many pager messages may be transmitted in a single call, so the paging server will continue to make as many separate calls as are required to process all of the page requests.

Page requests that are specified as urgent are sent immediately.

Pages may also be scheduled for future transmission. Messages waiting for their send time to arrive are sorted by send time and carrier. If there are multiple messages to send to the same carrier at the same time, an attempt will be made to send them all during a single connection, subject to the carrier-imposed limitations on connection time and message length.

Scheduled messages whose send time has arrived take priority over batch messages. A batch transmission that is in progress will be allowed to complete, at which time all messages scheduled for the current time will be sent. Further processing of any pending batch messages can then resume.

Page requests whose send time is in the past (either because that is what was supplied by the user or because upon return to service following a failure it is discovered that scheduled messages have not been sent) are treated as urgent and sent immediately.

Unsent pager messages are stored on disk so that if the CIC system is unavailable for some period of time the messages are maintained and sending is resumed upon the system's return to service.

Welcome to Interaction Designer

What is Interaction Designer?

Interaction Designer is CIC's (Customer Interaction Center) graphical application generator. Using Interaction Designer, you write the programs (called handlers) that control various interaction processing behaviors within CIC.

Use Interaction Designer to modify existing handlers and create new handlers. When you are ready to begin using the handler on an IC server, you must first publish and activate the handler.

Other Documentation

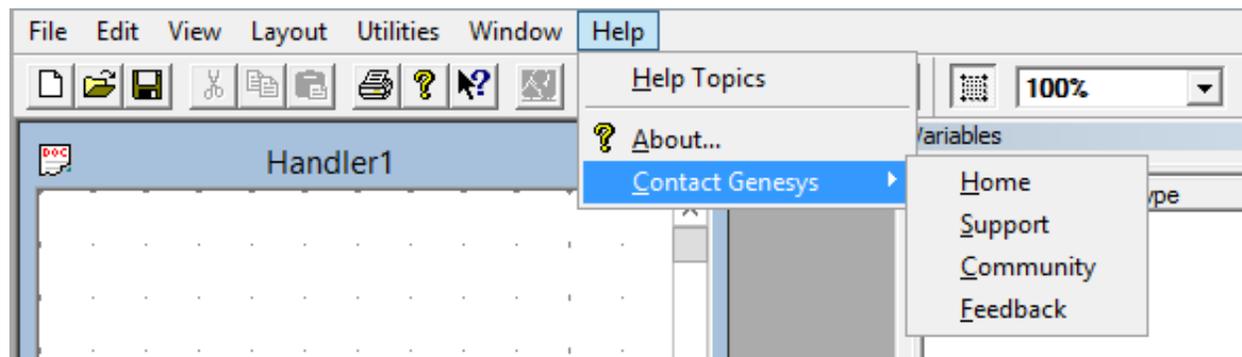
Be sure to explore the PureConnect Documentation Library on your CIC Server. This contains online help and technical reference documents that provide additional information and explanations of CIC features.

Note: See the help topic What's new in this release for information on new and changed tools, features, and handlers in each release.

Interact!

If you have questions or comments about this documentation, please send an email to PureConnectDoc@genesys.com.

To provide feedback on this program or product, and to find other resources that help you interact with Genesys staff and users, see the Help... **Contact Genesys** menus.



Incomplete Required Parameter dialog box

This dialog box appears when you try to click OK in the Properties notebook and you have not typed a value for a required field. The name of the empty field appears in the error message.

Insert Email Object Response Attachment

This tool was deprecated in CIC 4.0.

Integer to Call ID

This tool is no longer in the tool palette. It was replaced in version 2.4 with the [Convert String to Call Id](#) tool.

Integer to Conference ID

This tool is no longer in the tool palette. It was replaced in version 2.4 by the [Convert String to Conference ID](#) tool.

Introduction to Paging in CIC

The paging tools and initiators allow CIC users to send pages using handlers. When a user sends a page, CIC generates a Send notification for the object of Paging type. This starts either a handler containing the Send Pager Message Initiator, or if only the recipient name was specified, a handler containing the Send Pager Message Lookup initiator. In each of these handlers is a Send Pager Message tool that sends the page.

For information on configuring CIC's paging features, see the Interaction Administrator online help.

See [Sending Pages](#) for more information on Pager Server and the paging handlers.

See [How Paging Works in CIC](#) for more pager background information.

Interaction Client Users can specify the following parameters when sending a page:

- **Recipient name**
The name of a page recipient. This is not required if telephone number and pager id are specified. A page can only be sent with Recipient name if a handler is started with the Send Pager Message Lookup initiator. By default, there are no shipping handlers that use this initiator, so you cannot send a page by specifying Recipient name. You must use Carrier phone number and Pager ID.
- **Carrier phone number**
The telephone number for the paging service to which the pager is connected. For example, this might be SkyTel's carrier number.
- **Pager id**
The unique identifier for a pager. This is established by the company providing the paging services, and may be omitted if the recipient name is specified.
- **Message text**
The message to be sent. This may or may not be received, depending on the capabilities of the company providing the paging services, the recipient's service level, or the pager model.
- **Urgent flag**
An indicator that the message should be sent immediately, and not queued for sending at a later time.
- **Send date/time**
The date and time at which the message should be sent. This is not required if the message is marked urgent. If no date or time is specified, and the message is not marked urgent, the page is queued and sent at an interval specified in Interaction Administrator.

ISO 3166 Country Codes

This alphabetical list shows all countries and their corresponding two-letter country codes as per ISO 3166.

Afghanistan	AF
Albania	AL
Algeria	DZ
American Samoa	AS
Andorra	AD
Angola	AO

Antarctica	AQ
Antigua	AG
Argentina	AR
Armenia	AM
Australia	AU
Austria	AT
Azerbaijan	AZ
Bahamas	BS
Bahrain	BH
Bangladesh	BD
Barbados	BB
Belgium	BE
Belize	BZ
Benin	BJ
Bermuda	BM
Bhutan	BT
Bolivia	BO
Bosnia-Herzegovina	BA
Botswana	BW
Bouvet Island	BV
Brazil	BR
British Indian Ocean Territory	IO
British Virgin Islands	VG
Brunei	BN
Bulgaria	BG
Burma	BU
Burundi	BI
Byelorussian Ssr (Belorussia)	BY
United Republic of Cameroon	CM
Canada	CA
Canton and Enderbury Islands	CT
Cape Verde	CV
Cayman Islands	KY
Central African Republic	CF

Chad	TD
Chile	CL
China	CN
Christmas Island	CX
Cocos Islands	CC
Columbia	CO
Comoros	KM
Congo	CG
Cook Islands	CK
Costa Rica	CR
Cuba	CU
Cyprus	CY
Croatia	HR
Czechoslovakia	CS
Czech.	CZ
Denmark	DK
Djibouti	DJ
Dominica	DM
Dominican Republic	DO
Dronning Maud Land	NQ
East Timor	TP
Ecuador	EC
Egypt	EG
El Salvador	SV
Equatorial Guinea	GQ
Ethiopia	ET
Faeroe Islands	FO
Falkland Islands	FK
Fiji	FJ
Finland	FI
France	FR
French Guiana	GF
French Polynesia	PF
Gabon	GA

Gambia	GM
Federal Republic of Germany	DE
Georgia	GG
Ghana	GH
Gibraltar	GI
Greece	GR
Greenland	GL
Grenada	GD
Guadeloupe	GP
Guam	GU
Guatemala	GT
Guinea	GN
Guinea-Bisseu	GW
Guyana	GY
Haiti	HT
Heard and Mc Donald Islands	HM
Honduras	HN
Hong Kong	HK
Hungary	HU
Iceland	IS
India	IN
Indonesia	ID
Iran	IR
Iraq	IQ
Ireland	IE
Israel	IL
Italy	IT
Ivory Coast	CI
Jamaica	JM
Japan	JP
Johnston Island	JT
Jordan	JO
Democratic Kampuchea	KH
Kazakhstan	KK
Kenya	KE

Kiribati	KI
Democratic People's Rep. of Korea	KP
Republic of Korea	KR
Kuwait	KW
Kyrgyzstan (Kirgistan)	KG
Lao People's Democratic Republic	LA
Latvia	LV
Lebanon	LB
Lesotho	LS
Liberia	LR
Libyan Arab Jamahiriya	LY
Liechtenstein	LI
Lithuania	LT
Luxembourg	LU
Macau	MO
Madagascar	MG
Malawi	MW
Malasia	MY
Maldives	MV
Mali	ML
Malta	MT
Martinique	MQ
Mauritania	MR
Mauritius	MU
Mexico	MX
Midway Islands	MI
Moldova	MD
Monaco	MC
Mongolia	MN
Montserrat	MS
Morrocco	MA
Mozambique	MZ
Namibia	NA
Nauru	NR
Napal	NP

Netherlands	NL
Netherlands Antilles	AN
Neutral Zone	NT
New Calidonia	NC
New Zealand	NZ
Nicaragua	NI
Niger	NE
Nigeria	NG
Niue	NU
Norfolk Island	NF
Norway	NO
Oman	OM
Pacific Islands	PC
Pakistan	PK
Panama	PA
Papua New Guinea	PG
Paraguay	PY
Peru	PE
Phillipines	PH
Pitcairn Island	PN
Poland	PL
Portugal	PT
Puerto Rico	PR
Qatar	QA
Reunion	RE
Romania	RO
Russia	RU
Rwanda	RW
St. Helena	SH
St. Kitts Nevis Anguilla	KN
Saint Lucia	LC
St. Pierre and Miquelon	PM
Saint Vincent and the Grenadines	VC
Samoa	WS

San Marino	SM
Sao Tome and Principe	ST
Saudi Arabia	SA
Senegal	SN
Seychelles	SC
Sierra Leone	SL
Singapore	SG
Slovenia	SI
Slovakia	SQ
Solomon Islands	SB
Somalia	SO
South Africa	ZA
Spain	ES
Sri Lanka	LK
Sudan	SD
Suriname	SR
Svalbard and Jan Mayen Islands	SJ
Swaziland	SZ
Sweden	SE
Switzerland	CH
Syran Arab Republic	SY
Taiwan	TW
Tajikistan	TJ
United Republic of Tanzania	TZ
Thailand	TH
Togo	TG
Tokelau	TK
Tonga	TO
Trinidad and Tobago	TT
Tunisia	TN
Turkey	TR
Turkmenistan	TM
Turks and Caicos Islands	TC
Tuvalu	TV
Uganda	UG

Ukrainian SSR	UA
United Arab Emirates	AE
United Kingdom	GB
United States	US
United States Misc. Pacific Islands	PU
Unites States Virgin Islands	VI
Upper Volta	HV
Uruguay	UY
USSR	SU
Uzbekistan	UZ
Vanuatu	VU
Vatican City State	VA
Venezuela	VE
Vietnam	VN
Wake Island	WK
Wallis and Futuma Islands	WF
Western Sahara	EH
Yemen	YE
Democratic Yemen	YD
Yugoslavia	YU
Zaire	ZR
Zambia	ZM
Zimbabwe	ZW

ISO 639 Language Codes

This following is an alphabetical list of all languages along with their corresponding two-letter language codes as per ISO 639.

(Afan) Oromo	om
Abkhazian	ab
Afar	aa
Afrikaans	af
Albanian	sq
Amharic	am
Arabic	ar

Armenian	hy
Assamese	as
Aymara	ay
Azerbaijani	az
Bashkir	ba
Basque	eu
Bengali; Bangla	bn
Bhutani	dz
Bihari	bh
Bislama	bi
Breton	br
Bulgarian	bg
Burmese	my
Byelorussian	be
Cambodian	km
Catalan	ca
Chinese	zh
Corsican	co
Croatian	hr
Czech	cs
Danish	da
Dutch	nl
English	en
Esperanto	eo
Estonian	et
Faeroese	fo
Fiji	fj
Finnish	fi
French	fr
Frisian	fy
Galician	gl
Georgian	ka
German	de
Greek	el
Greenlandic	kl

Guarani	gn
Gujarati	gu
Hausa	ha
Hebrew	iw
Hindi	hi
Hungarian	hu
Icelandic	is
Indonesian	in
Interlingua	ia
Interlingue	ie
Inupiak	ik
Irish	ga
Italian	it
Japanese	ja
Javanese	jw
Kannada	kn
Kashmiri	ks
Kazakh	kk
Kinyarwanda	rw
Kirghiz	ky
Kirundi	rn
Korean	ko
Kurdish	ku
Laothian	lo
Latin	la
Latvian, Lettish	lv
Lingala	ln
Lithuanian	lt
Macedonian	mk
Malagasy	mg
Malay	ms
Malayalam	ml
Maltese	mt
Maori	mi
Marathi	mr

Moldavian	mo
Mongolian	mn
Nauru	na
Nepali	ne
Norwegian	no
Occitan	oc
Oriya	or
Pashto, Pushto	ps
Persian	fa
Polish	pl
Portuguese	pt
Punjabi	pa
Quechua	qu
Rhaeto-Romance	rm
Romanian	ro
Russian	ru
Samoan	sm
Sangro	sg
Sanskrit	sa
Scots Gaelic	gd
Serbian	sr
Serbo-Croatian	sh
Sesotho	st
Setswana	tn
Shona	sn
Sindhi	sd
Singhalese	si
Siswati	ss
Slovak	sk
Slovenian	sl
Somali	so
Spanish	es
Sundanese	su
Swahili	sw
Swedish	sv

Tagalog	tl
Tajik	tg
Tamil	ta
Tatar	tt
Tegulu	te
Thai	th
Tibetan	bo
Tigrinya	ti
Tonga	to
Tsonga	ts
Turkish	tr
Turkmen	tk
Twi	tw
Ukrainian	uk
Urdu	ur
Uzbek	uz
Vietnamese	vi
Volapuk	vo
Welsh	cy
Wolof	wo
Xhosa	xh
Yiddish	ji
Yoruba	yo
Zulu	zu

JPG file format for faxes

The [Create Fax Page List](#) and [Append Page tools](#) supports the following JPG file formats:

- JFIF. This is the JPEG File Interchange Format. CIC supports YUV 4:4:4, 4:2:2, and 4:1:1 color spacing, and YUV 4:0:0 for grayscale.
- Progressive JPEG. This is a JFIF format that is useful for transmitting images, because the first part of the file contains the full dimensions of the image. Therefore, in a paint-while-load routine, you can display the whole image, then progressively clarify it as the rest of the file loads. CIC supports YUV 4:4:4, 4:2:2, and 4:1:1 color spacing, and YUV 4:0:0 for grayscale.
- JTIF. This is the JPEG Tagged Interchange Format. CIC supports YUV 4:4:4, 4:2:2, and 4:1:1 color spacing, and YUV 4:0:0 for grayscale.

Key codes for use with the Host Press Key tool

Meaning	Mnemonic	Mainframe	AS/400
@	@@	X	X
Attention	@A@Q	X	X
Backspace	@<	X	X
Backtab (Left Tab)	@B	X	X
Clear	@C	X	X
Cmd (function) Key	@A@Y		X
Cursor Down	@V	X	X
Cursor Left	@L	X	X
Cursor Right	@Z	X	X
Cursor Up	@U	X	X
Delete	@D	X	X
Dup	@S@x	X	X
End	@q		X
Enter	@E	X	X
Erase EOF	@F	X	X
Erase EOL	@e@d		X
Erase Input	@A@F	X	X
Field Exit	@A@E		X
Field Mark	@S@y	X	
Field -	@A@-		X
Field +	@A@+		X
Help	@H		X
Hexadecimal	@A@X		X
Home	@0 (zero)	X	X
Insert	@I	X	X
Insert Toggle	@A@I		X
Local Print	@P		X
Left Tab (Back Tab)	@B	X	X
New Line	@e@n*	X	X
Page Up	@u		X
Page Down	@v		X
Print Screen	@A@T	X	

Reset	@R	X	X
Right Tab (Tab)	@T	X	X
Sys Request	@A@H	X	X
Tab (Right Tab)	@T	X	X
Test	@A@C		X
PA1	@x	X	X
PA2	@y	X	X
PA3	@z	X	X
PA4	@+	X	
PA5	@%	X	
PA6	@&	X	
PA7	@'	X	
PA8	@(X	
PA9	@)	X	
PA10	@*	X	
PF1/F1	@1	X	X
PF2/F2	@2	X	X
PF3/F3	@3	X	X
PF4/F4	@4	X	X
PF5/F5	@5	X	X
PF6/F6	@6	X	X
PF7/F7	@7	X	X
PF8/F8	@8	X	X
PF9/F9	@9	X	X
PF10/F10	@a	X	X
PF11/F11	@b	X	X
PF12/F12	@c	X	X
PF13/F13	@d	X	X
PF14/F14	@e	X	X
PF15/F15	@f	X	X
PF16/F16	@g	X	X
PF17/F17	@h	X	X
PF18/F18	@i	X	X
PF19/F19	@j	X	X

PF20/F20	@k	X	X
PF21/F21	@l	X	X
PF22/F22	@m	X	X
PF23/F23	@n	X	X
PF24/F24	@o	X	X

Keyword Spotting by Sets

This Telephony tool enables CIC to start or stop monitoring a call for key words. If it identifies key words that are defined and associated with the specified GUIDs for the customer and agent channels, it initiates notifications. In its most basic form, the tool stores the spotted key words and associates them with the call.

Note: Before you use this tool, you should check the keyword spotting feature in the Interaction Recorder Policy Editor. As of CIC 4.0 SU3, Policy Editor offers keyword spotting by workgroups, roles, individual users, lines, and stations. Use this tool in situations where keyword spotting in Policy Editor doesn't meet your needs.

Inputs

Call Identifier

The unique identifier of the call.

Action

Start to begin monitoring for key words, or Stop to end monitoring of a call. Start and Stop are the only supported actions.

Keyword Set GUIDs for Customer

A list of the GUIDs for the keyword sets to apply to the customer's channel.

Keyword Set GUIDs for Agent

A list of the GUIDs for the keyword sets to apply to the agent's channel.

Language

The language associated with the call.

Exit Paths

Success

This tool takes the Success exit path if the keyword spotting API call completed successfully.

Failure

This tool takes the Failure exit path if the API call failed or if the specified action is invalid.

Layout Toolbar

This menu contains options allowing you to control the appearance of the active handler window.

Layout Toolbar

Click	To
	Toggle the grid in the design window background.
	Change the current zoom level in the design window.
	Align all selected steps so that the left border of each icon is even with that of the primary selected step.
	Align all selected steps so that the top border of each icon is even with that of the primary selected step.
	Align all selected steps so that the bottom border of each icon is even with that of the primary selected step.
	Align all selected steps so that the right border of each icon is even with that of the primary selected step.

Listen

The Listen tool is similar to the Listen button on a CIC client toolbar. It allows one person to listen to another user's call without detection. This tool was written to enable Listen functionality for user's not running a CIC client.

For example, you could write a handler that allows you to enter a code and a station extension. A handler could then allow you to listen to the currently connected call on that station.

Inputs

Call Identifier

The call to listen to.

Station Queue Identifier

The station that will listen in on the call. For example, if a supervisor is listening to agent calls, the Station Queue Identifier would be the supervisor's station queue.

Continue Listening if call transferred? option

Select this option if you want to listen to the call after it is transferred. Clear this option if you want to stop listening when the call is transferred.

Exit Paths

Success

This tool takes the success path if the Listen operation succeeds.

Failure

This tool can take the Failure exit path for several reasons. Failure can occur if the call disconnects, if the call is picked up by an CIC client user (and is no longer under the control of the handler), the call ID is no longer valid (if the call is deallocated), or system resource limitation.

Literal Values

Literal values are the simplest of expressions. For example, if an Email Address input parameter takes a string value, you can type "joebob@aol.com". When the tool executes, the email is sent to the specified address.

Remember that literal string values must be placed within quotation marks "joebob@aol.com". See [Literal Value operators](#) for more information on syntax to use when typing literal values.

Using Literal Values in complex expressions

One of the more useful features of literal values is they can form part of a larger expression. For example, you might have a Play Audio step that looks for the Do Not Disturb recording for a user. Your expression might look something like this:

```
StrUserName & "_Do_Not_Disturb.wav"
```

This expression appends the string value contained in StrUserName to the literal text "_Do_Not_Disturb.wav" to form a unique filename when the tool executes.

Special Characters in Expressions

Expression Editor Assistant reserves (and interprets) some characters as part of an expression and not as part of a literal value. For example, if you want to use the " (quote) character in a string, you would use \". There are also special characters for inserting line breaks and tabs as part of a string value.

While you can use these special characters in a string, they require special syntax, typically a preceding \ character. See [Literal Value operators](#) for more information on special characters in literal values.

Related Topics

[Literal Values, Variables, and Operators](#)

[Operators](#)

[Variables](#)

MAC, IMG, and ISP file formats for faxing

The [Create Fax Page List](#) and [Append Page tools](#) supports the following 1-bit file formats:

- MacPaint (MAC): These Macintosh Paint files are commonly used for monochrome clip art.
- GEM Image (IMG): These files are native to the Graphical Environment Manager developed by Digital Research.
- Microsoft Paint (MSP): These files from early versions of Windows are used for black-and-white drawings and clip art.

Import Global Variables

Opens the Global Variable Manager dialog, where handler authors can import global variables into the current handlers that are stored on the CIC server.

When importing global variables, Interaction Designer ensures that the name of each variable being imported is unique. If a handler contains a variable that matches the global variable name and you try to import the variable, then Interaction Designer does the following:

- Lets you bind the local variable to the global variable if their data types are the same, or
- Assigns a unique name of a new local variable to the global variable.

Note: Because the global variable is locked throughout the duration of a handler instance, handler developers should write subroutines that read/modify the contents of global variables to minimize the time a global variable is locked.

Global variable values do not persist through restarts of Interaction Processor. Global variables are visible only within a single CIC server.

Manual Dial Request Initiator

This initiator begins when a person generates a manual dial request event by manually placing a call from a telephone (instead of using a CIC Client).

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Call

Object ID

Select {all}.

Notification Event

Select Manual Dial Request.

Outputs

Call Identifier

The identifier for the object that started this handler. A Call Identifier is a value that stays the same throughout the life of a call.

Workstation Queue Identifier

The name of the station queue on which this call originated.

User Queue Identifier

The User Queue of the person logged in to the workstation from which a call is manually placed.

Exit Paths

Start

This step always exits through the start path.

Mathematical Operators

The following operators perform math operations.

? + ?

`(Numeric or Integer) + (Numeric or Integer)`

Adds two numbers of the same type (Integer or Numeric). The result of this operation is a value of the same type as the numbers added.

Example: 1+1 results in a value of 2.

? - ?

(Numeric or Integer) - (Numeric or Integer)

Subtracts the value on the right from the value on the left. The two numbers must be of the same type (Integer or Numeric). The result of this operation is a value of the same type as the numbers subtracted.

Example: 2-1 results in a value of 1.

? * ?

(Numeric or Integer) * (Numeric or Integer)

Multiplies two numbers of the same type (Integer or Numeric). The result of this operation is a value of the same type as the numbers multiplied.

Example: 4*3 results in a value of 12.

? / ?

(Numeric or Integer) / (Numeric or Integer)

Divides the value on the right into the value on the left. The two numbers must be of the same type (Integer or Numeric). The result of this operation is a value of the same type as the numbers subtracted.

Example: 16/8 results in a value of 2.

? ^ ?

(Numeric or Integer) ^ (Numeric or Integer)

Raises the value of a Numeric or Integer to the power of a Numeric or Integer.

Example: 2 ^ 3 results in a value of 8.

Abs

Abs (Integer)

Returns the absolute value of an Integer.

Example: Abs(3-4) results in a value of 1.

Mod

Mod (Integer, Integer)

Returns the remainder after dividing the right integer into the left integer.

Example: Mod (5,2) results in a value of 1 because 2 goes in 5 twice with a remainder of one.

- ?

- (Numeric or Integer)

Multiplies a value by negative one. In other words, converts a positive number to a negative number, and converts a negative number to a positive number.

Example: -(5) results in a value of 5. -(14) results in a value of -14.

Melder - Request Audio Path

Description

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Melder

Object ID

Select {all}.

Notification Event

Description

Outputs

Local Call Identifier

Description

Source Site Identifier

Description

Sink Site Identifier

Description

Correlation String

Description

Is Audio Path In Use?

Description

Exit Paths

Start

This step always takes the Start exit path.

Melder - Setup Audio Path

Description

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Melder

Object ID

Select {all}.

Notification Event

Description

Outputs

Correlation String

Description

Routing Path

Description

Exit Paths

Start

This step always takes the Start exit path.

MP3 files

CIC does not support MP3 audio files at this time. This is because Dialogic does not currently support MP3 audio on its boards.

Open an existing handler or subroutine

Open an existing handler to modify the steps or properties of a step. You must [publish](#) the handler before any changes you make take effect on the CIC server.

1. From the File menu, choose Open.

or



Click the  button on the toolbar.

2. From the Open dialog box that appears, navigate to and select the handler you want to open and click OK. Multiple handlers may be selected.
3. The handler opens.

Operator Descriptions

From this topic, you can jump to descriptions of the operations and literal values available from the Expression Editor Assistant. Click on a category below for descriptions and examples of operators within that category.

[Comparison operations](#)

[Date/Time operations](#)

[Literal values](#)

[List operations](#)

[Mathematical operations](#)

[String operations](#)

[Type Conversions](#)

Related Topics

[Introduction to Expression Builder](#)

Overview of Buffer Tools

The buffer tools are used to manipulate buffers in a stack manner. Data is stored in the buffer and stored in a binary format.

These tools are used in conjunction with the WebSphere MQ tools. If you are not using the WebSphere MQ tools you may never use these tools.

Click on one of the following Buffer tools for more information on that tool:

[BufferGetDate](#)

[BufferGetInteger](#)

[BufferGetLength](#)

[BufferGetString](#)

[BufferHandleToInteger](#)

[BufferPutDate](#)

[BufferPutInteger](#)

[BufferPutString](#)

[CopyBuffer](#)

[CreateBuffer](#)

[DecodeBuffer](#)

[DeleteBuffer](#)

[EncodeBuffer](#)

[IntegerToBufferHandle](#)

Overview of Multi-Site Tools

The Multi-Site messaging tools can be used to construct an arbitrary message format within a handler and receive that message in another handler on a remote server. The format of the message is determined by the order in which the various elements are added to the message. On the receiving end, the elements must be read from the message in the correct order. The type of the element is checked when it is read, so an attempt to read the wrong element type results in an error.

When a message is ready to be sent, the sender specifies a destination site ID and the object and event values that are used to start the notification on the receiving server. Messages are sent asynchronously, requiring no response, or synchronously, where the sending tool waits a specified amount of time for a response message to come back. The response message is read in exactly the same way as the original message by using the *get* tools in the same order as the sender used the *put* tools.

See the white paper, *A Guide to Interaction Multi-Site*, for a more detailed explanation of Multi-Site messaging in CIC.

Click on a tool below to learn more about that tool:

[Multi-Site Create Message](#)

[Multi-Site Get Integer](#)

[Multi-Site Get Note](#)

[Multi-Site Get String](#)

[Multi-Site Put Integer](#)

[Multi-Site Put Note](#)

[Multi-Site Put String](#)

[Multi-Site Send Event](#)

[Multi-Site Send Request](#)

[Multi-Site Send Response](#)

Palette Toolbar

Click	To
	Open or close the Design palette.
 , or 	Open or close the messages bar. Whether the message bar is open or not, the button will display as a red frown if the handler is not in a state that will allow it to be published (i.e., contains steps with required fields not yet assigned values). This button will display as a yellow grim face or a green smile whenever the handler is in a state that can be published. The yellow face indicates that the handler is in a publishable state, but that the message log contains errors. The green smiling face indicates a publishable handler without any errors in the message log. Note that this does not necessarily mean that the handler will function as intended, merely that all required fields have been assigned values and the handler is in a state that can be published.
	Open or close the Variables palette.
	Open or close the QuickJump palette.
	Open or close the Debug palette.

PCD and FPX file formats for faxes

The [Create Fax Page List](#) and [Append Page tools](#) supports the following Kodak file formats:

- PhotoCD (PCD). This is a common, high-resolution format for images on CD-ROM. An image file can contain one or more physical resolutions, with fixed dimensions. You can choose which resolution to load, and thus control the size of the resulting bitmap.
- FlashPix (FPX). This Kodak format is newer than PhotoCD. This format can also contain more than one physical resolution, but without fixed dimensions. You can choose which resolution to load, and thus control the size of the resulting bitmap.

For PCD and FPX files, you can read the following bits per pixel: 8, 24.

PCT file formats for faxing

The [Create Fax Page List](#) and [Append Page tools](#) supports the following PCT file formats:

- Macintosh Pict (PCT) files are produced using Macintosh QuickDraw, and are used in desktop publishing and imaging applications.

For PCT files, you can read the following bits per pixel: 1, 4, 8, 24.

PCX file formats for faxes and OCR

The [Create Fax Page List](#), [Append Page tools](#), and [OCR for TIFF/PCX/DCX files](#) supports the following PCX file formats:

- PCX. This is a file format created by ZSoft. This format compresses its image data with the RLE type compression.
- DCX. This is a multipage PCX format that enables a file to contain more than one image. It is handled the same as a regular PCX file, except for the multipage feature.

For PCX and DCX files, you can read the following bits per pixel: 1, 4, 8, 24.

PNG file formats for faxing

The [Create Fax Page List](#) and [Append Page tools](#) supports the following PNG file formats:

- PNG: PNG (Portable Network Graphics) is a replacement for the GIF format. It is a full-featured (non-LZW) compressed format intended for widespread use without legal restraints.

You can read the following bits per pixel: 1, 4, 8, 16, 24, 32.

Power Tools command (Utilities menu)

Use this command to open the list of power tools. Power tools are extensions to the standard Interaction Designer functionality and are intended for veteran handler developers. The power tools allow you to view handler differences, auto-label steps, view XML code for individual steps, and more.

[Auto Label Assignments](#)

[Auto Label Conditions](#)

[Handler Diff](#)

[Relabel Assignments](#)

[Relabel Conditions](#)

[Toolstep Info](#)

[Step XML](#)

Predictive Call Completed

An outgoing predictive call has completed. Return the reason and finish code for the completion of the call to the Dialer Server. (Mainly used in our default handler to communicate Busy, No Answer, or SIT calls).

Caution: Do not use this tool in your handlers as it may cause CIC to function incorrectly. It is intended for Interaction Director handlers only. You will never need to use or modify the values set in this tool.

Inputs

Campaign ID

Campaign identifier. The name of the current campaign with which the call is associated.

Request ID

Internal Dialer Id representing the campaign request that contains the current call. (A request can have multiple calls associated with it.)

Reference ID

Internal Dialer Id representing the call itself.

Call Completion Code

Dialer Code representing the reason why the call ended.

Finish Code

User-defined descriptive term representing the reason why the call failed.

Exit Paths

Success

This path is taken if the call information is successfully returned.

Failure

This path is taken if the operation fails.

Preview Call

Send a Preview Call (Call Data) to a Dialer Agent. A preview dialing call is when the data for a call is first routed to an agent where they read through and become familiar with who they are calling and then they physically place the call. The preview call is simply data representing the call that is routed to an agent.

Caution: Do not use this tool in your handlers as it may cause CIC to function incorrectly. It is intended for Interaction Director handlers only. You will never need to use or modify the values set in this tool.

Inputs

Predictive Attribute Name List

List of attribute names for the call.

Predictive Attribute Value List

List of attribute values for the call.

Queue name

Agent Id that the preview call will be routed to.

Exit Paths

Success

This path is taken if the requested call data is successfully sent.

Failure

This path is taken if the operation fails.

Print a handler

Using this feature, you can print a handler at several magnifications. For example, you can print an entire handler on one page, or you can print a larger view of a handler across several pages.

1. From the File menu, click Print.

or



Click the button on the toolbar.

The Print dialog box appears.

2. From the Print dialog box, make any necessary configurations, and then click OK.

Print Magnification

Size Scale is the magnification of the printout. The higher the Size Scale number, the larger your image will be.

Printer Layout Options

Layout options allow you to print your handler or subroutine to one page or across multiple pages.

Printing a handler across multiple pages

Click Multi Pages to print the handler or subroutine across multiple pages.

Printing a handler on a single page

Click All to print the entire handler or subroutine on one page.

Properties

This is a text box that allows handler authors to write notes about the handler in the active window. Handler authors may use this to write a general description of the handler as a whole or to provide specific information that another user might need to know when working with the handler at a later time.

PSD file formats for faxing

The [Create Fax Page List](#) and [Append Page tools](#) supports the following PSD file formats:

- PSD: Photoshop (PSD) is the format produced by the Adobe Photoshop graphics editor.

For PSD files, you can read the following bits per pixel: 1, 8, 24.

Publish Command (File Menu)

Use this command to publish a handler to the CIC server. For more information on publishing, see [Publish a handler or subroutine](#).

General Preferences

This page can be used to set three general defaults for Interaction Designer.

Publish Handler Dialog

If this box is checked, users will be prompted to continue any time they try to publish a handler that contains warnings. If unchecked, the handler will publish without prompting even though it may contain warnings.

Starting Directory for File Open and Save As

This field can be used to set a default directory for Open or Save As dialog. If left undefined, ID will start at the last directory that was accessed for opening or saving a handler.

DB Query Step Database Operations

These settings are used by the [DB Query](#) tool.

Timeout for datasource retrieval

The number of seconds that the tool will wait to retrieve datasources displayed in the Data Source dropdown listbox. Setting the timeout period to zero will cause the tool to wait indefinitely.

Include synonyms in the Table listbox

If this box is checked, synonyms will be included in the Table Name listbox. Including synonyms can increase the amount of time needed for the query, so some users may not want to include them.

Related Topics

[Debugging Preferences](#)

[Designer Preferences Page](#)

[Handler Download Preferences](#)

[Views Preferences](#)

Publishing IC 2.2 handlers in IC 2.3

If you wrote or modified IC 2.2 handlers and publish them in IC 2.3, you may receive Interaction Designer error messages indicating that a tool's parameters and/or exit paths have changed since the previous release. This error message will tell you the offending tool's Step Label and Node ID, and attempt to explain the problem with an error message.

You cannot publish this handler until you correct the problem. Use the Step Label, Node ID, and error message text to locate the step and diagnose the problem. Make any changes that are necessary to solve the problem, such as creating a variable to contain the value of the parameter, or creating a link from the new exit path. Then save the handler and try publishing it again.

If you are still unable to solve the problem, contact technical support.

Publishing IC 2.2 handlers in IC 2.4x

If you wrote or modified IC 2.2 handlers, you must first save them in IC 2.3x or greater before you can publish them in IC 2.4x. Depending on the content of the original handler, you may receive Interaction Designer error messages indicating that a tool's parameters and/or exit paths have changed since the previous release. This error message will tell you the offending tool's Step Label and Node ID, and attempt to explain the problem with an error message.

You cannot publish this handler until you correct the problem and save it in the newer version. Use the Step Label, Node ID, and error message text to locate the step and diagnose the problem. Make any changes that are necessary to solve the problem, such as creating a variable to contain the value of the parameter, or creating a link from the new exit path. Then save the handler and try publishing it again.

If you are still unable to solve the problem, contact technical support.

QuickJump

The QuickJump Palette allows you to easily locate a particular step within the open handler. There are four view options for the QuickJump palette, selectable through the Utilities menu. Each view identifies the step by label, type, and ID. Double-click on a step listed in the QuickJump palette to move focus to that tool in the design window.

List all steps

This will list all steps in the open handler in the QuickJump palette.

List Steps With No Incoming Links

Only steps with no incoming links will appear in the QuickJump palette.

List Steps With Unlinked Exit Paths

Shows all steps with any unlinked exit paths. This is useful in troubleshooting the handler's exit points.

Clear QuickJump List

This will remove all steps from the QuickJump palette. Select one of the above three options to repopulate the palette.

RAS file formats for faxing

The [Create Fax Page List](#) and [Append Page tools](#) support the following RAS file formats:

- SUN Raster (RAS) is a format native to Sun UNIX platforms.
For RAS files, you can read the following bits per pixel: 1, 4, 8, 24, 32.

Record Email Object

This tool was deprecated in CIC 4.0.

Register Call Data

This Dialer tool is not longer in the tool palette. It was previously used to associate an incoming call with a campaign list entry and return the detailed data for the caller from the Interaction Dialer Campaign COM object.

Inputs

Campaign ID

Campaign identifier. The name of the current campaign that this call is associated with.

Request ID

Internal Dialer Id representing the campaign request that contains the current call. (A request can have multiple calls associated with it).

Reference ID

Internal Dialer Id representing the call itself.

Call ID

Call Id for the outbound call

Predictive Attribute Name List

List of attribute names for the call.

Predictive Attribute Value List

List of attribute values for the call.

Exit Paths

Success

Key used to look up this person in the campaign database. The phone number is the unique key that identifies this caller and will be used to look up their detailed data from the database

Failure

The name of the table to which the attributes are written.

Relabel Power Tools

Two Relabel power tools are available in Interaction Designer for advanced users.

If you select a toolstep, or multiple toolsteps, and then click the Relabel tool, the step label changes to the contents of the step. You might have to expand the step to see the entire label.

- The **Relabel Assignments** tool affects the assignment tool steps in a handler.
- The **Relabel Conditions** tool affects the condition tool steps in a handler.

To enable one of these tools:

- From the **Utilities** menu, point to **Power Tools**, and then choose either the **Relabel Assignments** or **Relabel Conditions** tool.

See Also

[Auto Label Power Tools](#)

Remove Email Object Response Attachment

This tool was deprecated in CIC 4.0.

Sample.lst

A custom handler list is just a list of handler names saved with the .lst extension. Here's an example:

```
handler1.ihd  
handler2.ihd  
handler3.ihd  
handler4.ihd  
handler5.ihd  
handler6.ihd
```

Select All

Selects all tools in the current window.

Send Message Extended

This SMDI tool was deprecated in 4.0.

Send Pager Message Initiator

Send Pager Message tells the CIC Paging Server to send a page to a TAP compatible paging service. This initiator starts the PagingDefault handler. Send Pager Message Initiator starts when CIC generates a Send notification for a Paging object. This occurs when someone sends a page specifying only the Carrier Phone Number and Pager ID. PagingDefault.ihd uses this initiator.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Paging

Object ID

{all}

Notification Event

Send

Outputs

Carrier Phone Number

The telephone number for the paging service to which the pager is connected.

Pager ID

The unique identifier for a pager. This is established by the company providing the paging services, and may be omitted if the recipient name is specified.

Message Text

The message to be sent. This may or may not be received, depending on the capabilities of the company providing the paging services, the recipient's service level, or the pager model.

Urgent

An indicator that the message should be sent immediately, and not queued for sending later.

Send Date/Time

The date and time when the message should be sent. This is not required if the message is marked Urgent.

Exit Paths

Start

This step always exits through the start path.

Send Pager Message Lookup Initiator

This initiator starts when CIC generates a `SendWithLookup` notification for a Paging object. This occurs when someone sends a page from Paging Client specifying only the recipient's name. Send Pager Message Lookup Initiator doesn't start any handlers. This initiator starts when CIC generates a `SendWithLookup` notification for a Paging object. This occurs when someone sends a page specifying only the recipient's name.

Note: This initiator is not currently supported in the CIC handlers, so you cannot send a page by specifying the recipient name. You must specify the carrier phone number and pager Id instead, and use the `PagingDefault.ihd` handler.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Paging

Object ID

{all}

Notification Event

`SendWithLookup`

Outputs

Recipient Name

The name of the page recipient.

Message Text

The message to be sent. This may or may not be received, depending on the capabilities of the company providing the paging services, the recipients service level, or the pager model.

Urgent

An indicator that the message should be sent immediately, and not queued for sending later.

Send Date/Time

The date and time when the message should be sent. This is not required if the message is marked urgent.

Exit Paths

Start

This step always exits through the start path.

Send Email Object Response

This tool was deprecated in CIC 4.0.

Set MWI Extended 2

This SMDI tool was deprecated in 4.0.

Set MWI Extended

This SMDI tool was deprecated in 4.0.

SMDI : Send Message

This SMDI tool was deprecated in 4.0.

SOAP Wizard

SOAP stands for **Simple Object Access Protocol**. SOAP is a protocol for exchanging XML-based messages over computer networks, normally using HTTP/HTTPS.

For example, the SOAP Listener task on an IIS server uses HTTP protocol to transport SOAP messages to and from the Internet. Applications developed using SOAP Notifier COM components use Notifier protocol to transport SOAP messages to and from a CIC server. SOAP itself is not concerned with the protocol used for transport, so you can use SOAP on many types of computer networks.

The SOAP tools allow you to build a SOAP Request subroutine. The Proxy wizard creates a handler that exposes a SOAP interface for an existing subroutine initiator. You use the wizard to define the calling subroutine and then the variables to use. With this information, the Proxy wizard builds a subroutine you can use with the SOAP tools.

SOAP Tool Variables

The SOAP tools use only variables of the type "string" or "list of strings," so any subroutine to which you are going to add a SOAP request must contain only those types of variables.

Web Services Invoked by SOAP

A web service is a method that you invoke across the Internet. A web service can perform virtually any data processing activity, ranging from simple information lookups to complicated business transactions. SOAP is frequently used to invoke web services. In the SOAP proxy wizard, you must enter a web service name with an IIS server host name and port.

ISAPI Listener Files

These files are used on the web server. The SOAP Listener task on an IIS server uses HTTP protocol to transport SOAP messages to and from the Internet. These files include the Web Services Description file and the ISAPI Filter Config file. The Web Services Description file makes it possible to describe a service on the CIC server so that a worldwide audience can find and use it. SOAP clients use this file to generate code to communicate with the web service. The ISAPI Filter Config file filters incoming message requests. This file prevents denial of service attacks.

Note: For more installation and configuration information, consult the *Installing and Using SOAP Functionality Technical Reference and Installation Guide*.

Proxy Handler Options

These options define how the proxy wizard generates specific logic in the handlers that it generates.

Failure Handling Logic

You cannot define exit paths for a subroutine initiator. A subroutine can use output parameters to signify a failure. Check this option to add failure handling tool step logic.

If you choose this option, when the handler is created, two tool steps are added to help with this. These two tool steps will have TODO in the tool step label. You will have to change the expression, which defaults to true, of the condition tool step and set the fault string, which defaults to empty string, in the assignment step.

Copy Header

If you want to copy the <Header> element and its contents from the source envelope to the response envelope, select this option. This will affect both success and failure paths.

Note: For more information, refer to the *Installing and Using SOAP Functionality Technical Reference and Installation Guide*.

See Also

Using the SOAP Proxy Wizard

Special Export Options

The following is a table of the export types, and the special export options that can be used with these export types. (Separate multiple special options with a semicolon, i.e., "ExcelColumnHeadings=1;ExcelConstColWidth8.")

Export Types	Special Parameters Supported
Microsoft Excel 97-2000 Extended	<p>ExcelConstColWidth Uses a constant column width of 36 points.</p> <p>ExcelConstColWidth=NNN.N Set constant column size in points. Value can be a floating decimal. There is no default.</p> <p>ExcelBaseAreaType=(WHOLEREPORT DETAIL PAGEFOOTER PAGEHEADER GROUPHEADER GROUPFOOTER) Section type used to define column alignment and widths. Default is WHOLEREPORT.</p> <p>ExcelBaseAreaGroupNum=N Valid only for GROUPHEADER or GROUPFOOTER and identifies the grouping to use. Defaults to 1 if no value is specified.</p> <p>ExcelExportPageBreaks Export page breaks. Default is false.</p> <p>ExcelConvertDateValuesToString Convert date time values. Default is false.</p> <p>ExcelNoExportPageHeaders Do not export page headers. Default is false (report will export page headers).</p> <p>FirstPageNo=NNN First page of a page range to export. If not specified, all pages are exported.</p> <p>LastPageNo=NNN Last page of a page range to export. If not specified, the last page is used by default.</p>
Paginated Text	LinesPerPage=n
Text, Tabbed Text, Comma Separated Text, Tab Separated Text, Character Separated Text, Records, DIF	<p>UseReportNumberFormat=0 or 1</p> <p>UseReportDateFormat=0 or 1</p>
Character Separated Text	<p>StringDelimiter={character}</p> <p>FieldDelimiter={character}</p>
<p>ODBC</p> <p>Note: The ODBC export does not do a good job formatting the exported data. For example, Crystal Reports generates column names from the report (and usually mangles them. Also, the report is exported sideways so the report title appears in every row on the left. We recommend that you export to some other type, such as a comma separated file, and import the data from that file.</p>	<p>DSN={ODBC data source} *required</p> <p>TABLE={export table} *required</p> <p>UID={logging in user id}</p> <p>PWD={logging in password}</p>

Explanation of Special Options

Any special option that takes a value must be followed immediately by an "=" and the values. Example ExcelConstColWidth=15. If there are any spaces, tabs, or other characters between the "=" on either side, the parameters is considered invalid.

ExcelConstColWidth or ExcelConstColWidth=1 to n

Default is off/false. If specified without a value constant column widths will be enabled with a default width of 36. If a non zero width is specified, then that width will be used to determine the fixed width of the columns.

ExcelBaseAreaType

Defines the base area to be used for tabular formatting. This will be some section of the Crystal reporting hierarchy. The possible values are WHOLEREPORT, DETAIL, PAGEHEADER, PAGEFOOTER, GROUPHEADER, and GROUPFOOTER.

If no ExcelBaseAreaType is defined, the WHOLEREPORT section will be used to define column alignment and widths.

If the base area is GROUPHEADER or GROUPFOOTER, the ExcelBaseAreaGroupNum identifies which of the group sections defines the tabular base area.

ExcelBaseAreaGroupNum=1 to n

If the ExcelBaseAreaType is GROUPHEADER or GROUPFOOTER, this defines the group that is to be used for the tabular format.

LinesPerPage=1 to n

For paginated export type, this sets the number of lines to be used for each page of output. The default value is 66.

UseReportNumberFormat=0 or 1

The default value for this is false. If a value of 1 (true) is supplied, the report number formatting will be used for numeric output values in the export. If the value is 0 (false), then the system default numeric formats as defined by the Locale settings will be used.

UseReportDateFormat=0 or 1

The default value for this is false. If a value of 1 (true) is supplied, the report date and time formatting will be used for date and time output values in the export. If the value is 0 (false), then the system default date and time formats as defined by the Locale settings will be used.

StringDelimiter={character}

Specified the single character that will be used as a delimiter at the beginning and end of a string value. The default is a double quote.

FieldDelimiter={character}

Specifies the single character that will be used to separate fields in the export. The default value is a comma.

DSN={ODBC data source}

Specified the ODBC data source name that will be used to connect to the database where the ODBC export should occur. This value is not optional for ODBC exports. If not provided the export will fail.

UID={logging in user id}

User identifier for the data source specified in the DSN export options. This value might not be required if the ODBC source has some other means of authenticating and connecting the user.

PWD={logging in password}

User password to be used with the UID and DSN. This value might not be required if the ODBC source has some other means of authenticating the connecting user.

TABLE={export table}

Destination table for the ODBC exports. This value is required for the export to succeed.

Standard Toolbar

Click	To
	Open a new handler.
	Open an existing handler or subroutine. Interaction Designer displays the Open dialog box, in which you can locate and open the desired handler or subroutine.
	Save the active handler or subroutine with its current name. If you have not named the handler or subroutine, Interaction Designer displays the Save As dialog box.
	Remove selected data from the document and stores it on the Clipboard.
	Copy the selection to the Clipboard.
	Insert the contents of the Clipboard at the insertion point.
	Print the active document.
	Displays information about Interaction Designer, including the version number.
	Opens the help topic for the selected item.
	Publish the current handler. This button is not available until all required fields have been assigned values.
	Initiates debugging.

Status Indicator

This icon indicates whether or not a handler is ready to publish. Clicking on this button opens and closes the [Messages Bar](#).

Icon	Meaning
	The handler can be published.
	The handler can be published, but there are warnings in the handler's message window.
	The handler cannot be published.

Step Variables

When execution is paused on a step, users can click on the Step Variables tab in the Debug Palette and Designer will show the variable values for all variables used by the currently displayed executing step.

Related Topics

[Call Stack](#)

[Debug Palette](#)

[Handler Variables](#)

[View the value of a variable in a debug handler](#)

[Watch Variables](#)

Step XML Power Tool

Interaction Designer can translate any tool step code into HTML. This is useful for easily viewing all the information about a step, including label information and expressions contained in the step. You can use the Export to XML command, but that tool exports the entire handler to XML and it can be difficult to find a specific step. The Step XML power tool displays the specific XML for any step you select.

To view the XML for a tool step:

1. From the **Utilities** menu, click **Power Tools**, and then click **Step XML**.

The Step XML dialog appears.

2. In an open handler, select the step or steps you want to view in XML format.
3. In the Step XML dialog, click **Get Current Steps**.

An XML version of the tool step appears.

4. Do one of the following:
 - To copy the step XML to the clipboard, click **Copy to Clipboard**.
 - To keep the step XML window on top of all other windows, select the **On Top** check box.

STID

A STID is a Statistics Tracking Id assigned to a call. Calls are assigned to Statistics Groups with the Assign Stats Group tool. Stats groups allow Queue Manager to generate statistics about calls across queues.

Subroutine Parameter dialog box

From the Subroutine Parameter dialog box you can add new, or edit existing, subroutine parameters. The parameters you create in this initiator become inputs for the subroutine tool created when you publish this subroutine.

Parameter Label

The label that appears in the subroutine tool created when this subroutine is published.

Variable Name

The variable that contains the value passed in from the subroutine step that calls this subroutine.

Type

The data type of the variable. For example, the type might be String or Integer.

Option: This parameter will only be used as an input value to this subroutine

If you check this option, then this parameter will appear on the Inputs Page of the properties of any subroutine step that calls this subroutine. If you don't check this option, then the parameter will appear on the Outputs Page of the properties.

Default Value for Caller

This is an optional field where you can type a default value to appear for this parameter in the properties of the subroutine step that calls this subroutine.

Default Entry

The constant value or variable name to appear in the subroutine tool that calls this subroutine.

Option: Constant Value

Check this option if you want the value from the Default Value to be a constant value.

Option: Variable Name

Check this option if the value from the Default Value field is a variable name.

, symbol

A "," (comma symbol) can be used in a dialstring to indicate a two second pause. All of the Telephony tools accept this symbol.

/ symbol

A "/" (forward slash symbol) can be used in a dial string to indicate that numbers after the slash are to be dialed after a connection is made. All of the Telephony tools accept this symbol, which is used to support direct extension dialing.

TextFormat codes for ODK Export

You may specify the following codes in TextFormat= portion of your <Export> statement or in the Text Format input parameter.

TXT

Text

Word

Word7

MSWord

RTF

WordStar

WordPerfect

AmiPro20

AmiPro30

Ansi

Ascii

DatabaseAscii

DCA

DecolAscii

EBCDIC

Excel

FrameMaker

Interleaf

Lotus123WK1

Lotus123WK3

Lotus123WK4

LotusManuscript2x

MicrosoftWord40

MicrosoftWord5x

MicrosoftWord60

Multimate33

MultimateAdv36

MultiMateAdv37

OfficeWriter6x

PageMaker

PDAFormat

PFSFirstChoice20

PFSFirstChoice30

PFSProfWrite2x

Quattro

RichTextFormat

SamnaWordIVPlus

Ventura

WindowsWrite3x

WordForWindows1x
 WordForWindows2x
 WordForWindows60
 WordForWindows70
 WordPerfect42
 WordPerfect50
 WordPerfect51
 WordPerfect52
 WordPerfect60
 WordStar50
 WordStar55
 WordStar60
 WordStar70
 WordStarWindows1x
 XyWriteIIIPlus

TGA file formats for faxing

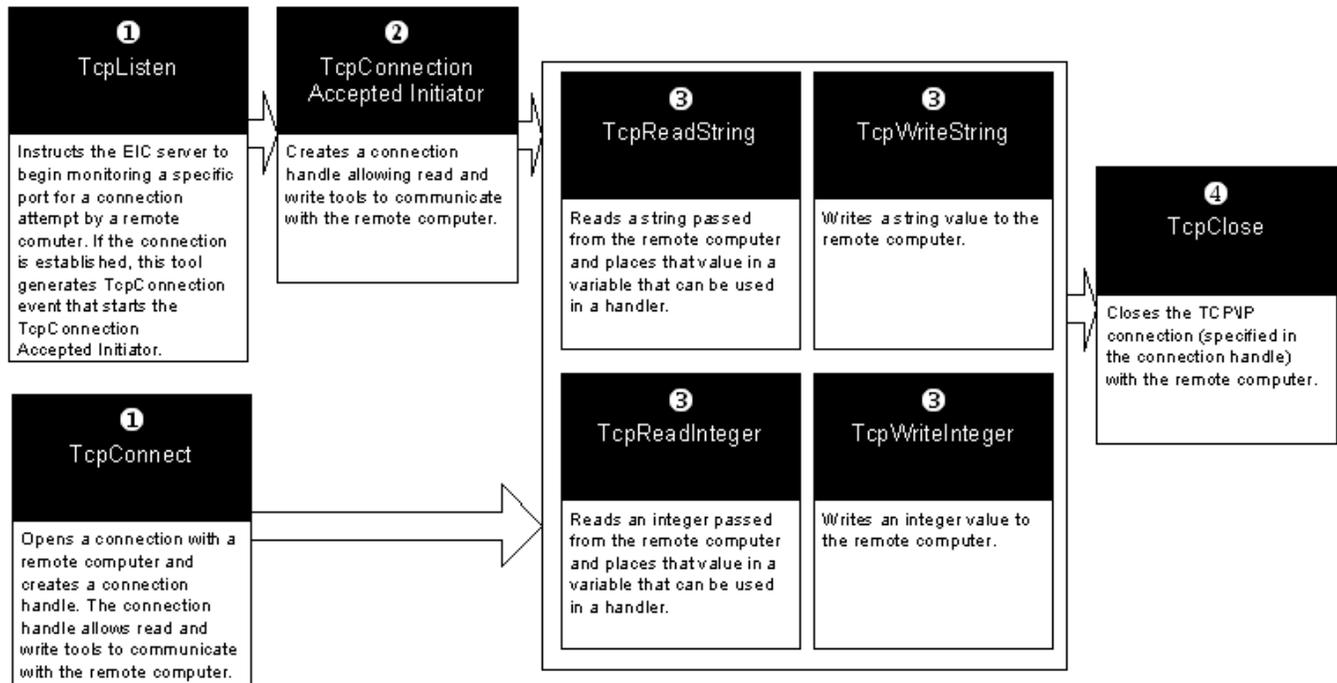
The [Create Fax Page List](#) and [Append Page tools](#) supports the following TGA file formats:

- TGA: Truevision TGA (TARGA) is a file format created by Truevision Inc. CIC supports all uncompressed and RLE compressed TGA file formats.

For TGA files, you can read the following bits per pixel: 8, 16, 24, 32.

The order in which TCP/IP tools should be used

This diagram suggests an order in which the [TCP/IP Tools](#) should be used. For best results when printing, set your paper orientation to Landscape.



TIF file format for faxes and OCR

The [Create Fax Page List](#) and [Append Page tools](#), supports the following TIFF file formats:

- TIFF. This is a tag-based file format designed to promote universal interchanges of digital image data. Because TIFF files do not have a single way to store image data, there are many versions of TIFF. [Create Fax Page List](#) supports the most common TIFF formats. CIC supports Group 3 and 4.
- MPT. This is a multipage TIFF format that enables a file to contain more than one image. It is handled the same as a regular TIFF file, except for the multipage feature.

You can read the following bits per pixel: 1, 2, 3, 4, 5, 6, 7, 8, 16, 24, 32.

Note: CIC does not support LZW compressed TIF files.

The [OCR for TIFF/PCX/DCX](#) tool supports the following formats:

- CCITT Group 3, 1D Compressed
- CCITT Group 4, 2D Compressed
- Tiff Uncompressed

OCR supports multi-page Tiffs, but they must all have the same resolution. For multi-page Tiffs, the resolution of the first page is used for all following pages. Minimum supported resolution is 100 dpi (dots per inch), maximum is 1200 dpi. 200 dpi is the minimum recommended resolution for accurate OCR processing.

SMDI: Set MWI

Turns on or off the message waiting indicator on a PBX phone. This tool will not work with non-PBX telephones.

Inputs

User Name

The User Name associated with the extension that will have its message waiting indicator turned on or off. User Names are defined for each CIC user in Interaction Administrator, and are an [attribute that can be looked up in Directory Services](#).

Turn on MWI

A value of true turns on the message waiting indicator on. A value of false turns the message waiting indicator off.

Exit Paths

Next

This tool always takes the Next exit path.

Toggle Messages

Opens or closes the Messages toolbar.

Tones

Both the Play Tone and Extended Get Key tools generate or listen for tones. The following tables show the parameters for some of the more common tones you can listen for or generate.

Note: Because the Play Tone tool does not have an Off Time parameter, you will not be able to generate Busy, Congestion, Reorder, or Ringback tones. To play these tones, you should create a .WAV file containing the tone and play it to call.

The length information in the following table is listed in *milliseconds*. If you are using these values for the [Play Tone](#) tool, you must convert these values to *seconds* (divide the millisecond value by 1000).

The following table shows the parameters for busy signals, congestion, reorder, and ringback.

Tone	Frequency 1	Frequency 2	On Time	Off Time
Busy	480	620	500	500
Congestion (Toll)	480	620	200	300
Reorder (Local)	480	620	300	200
Ringback	440	480	2000	4000

The following table shows the DTMF Tone Specifications. These are the same tones generated when a key is pressed on a telephone keypad.

Code	Tone Pair Frequencies (Hz)	Default Length (ms)
1	697, 1209	100
2	697, 1336	100
3	697, 1477	100
4	770, 1209	100
5	770, 1336	100
6	770, 1477	100
7	852, 1209	100
8	852, 1336	100
9	852, 1477	100
0	941, 1336	100
*	941, 1209	100
#	941, 1477	100
a	697, 1633	100
b	770, 1633	100
c	852, 1633	100
d	941, 1633	100

Toolstep Info Power Tool

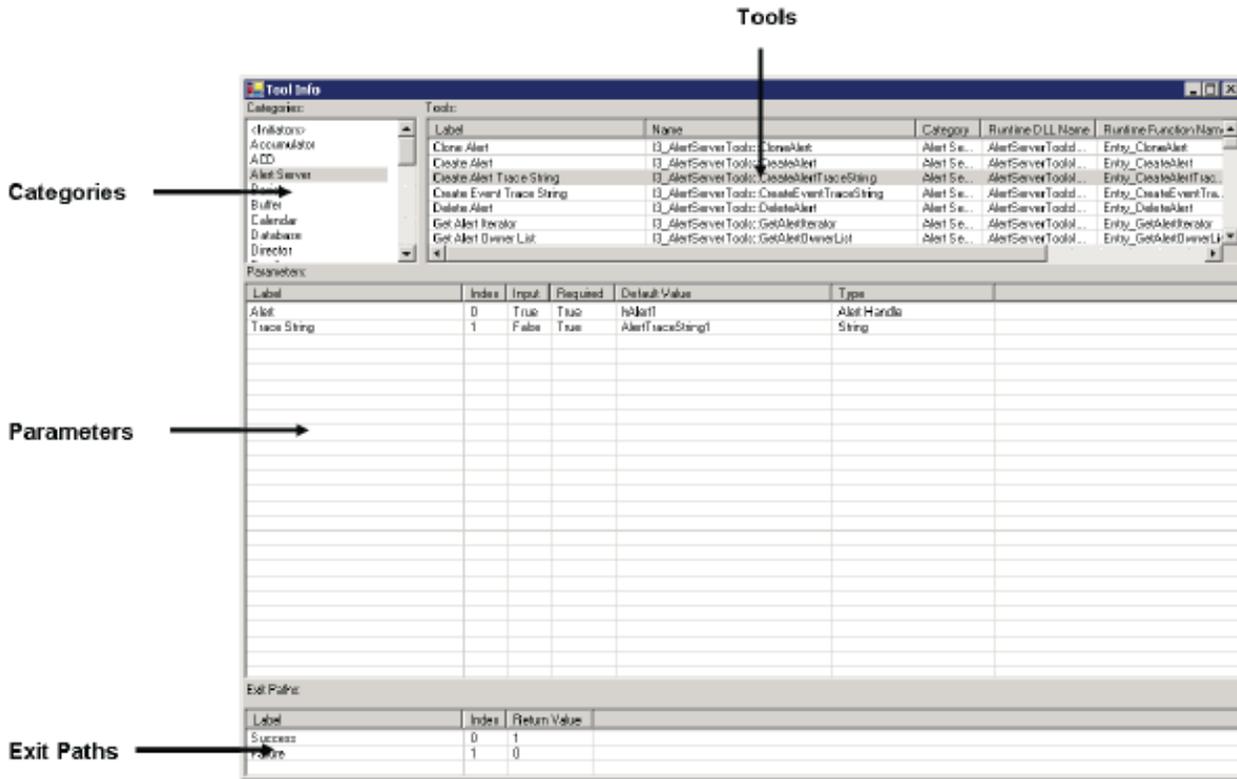
Each tool belongs to a category and is comprised of a number of parameters that have specific types. The Tool Info dialog allows you to see this information along with other information about the tool parameters. This power tool was used to write the SOAP wizard. When writing code that generates handlers, you have to use the internal name of the tool step. The Toolstep Info power tool opens the Tool info window.

To view tool information:

1. To access this power tool, from the **Utilities** menu, point to **Power Tools**, and then click **Toolstep Info**.

2. Select a category from the **Categories** list.
3. Select a tool from the **Tools** list.

Information about the tool appears in the **Tool Info** window.



- The **Categories** section of the Tool Info dialog lists all the tool categories in alphabetical order, except for initiators, which are unique and therefore listed first. Select a category to choose tools from that category.
- The **Tools** section lists all the tools in a selected category in alphabetical order. The columns in this section are:

Column	Description
Label	The label of the tool that appears in Interaction Designer.
Name	The internal name of the tool.
Category	The tool's category.
Runtime DLL Name	The runtime .dll file that contains the information for the tool.
Runtime Function Name	The runtime function name called for the tool.
Version	The version, if defined, for the tool.
Help Context	The help context ID, if defined, for the tool.
Help File	The help file the help context ID refers to, if defined, for the tool.

- The **Parameters** section displays the parameters for the selected tool in the order in which they appear on the tool step. The columns in this section are:

Column	Description
Label	The label of the parameter as it appears on the tool step.
Index	Ordinal position of the parameter.
Input	If the parameter is required, this is set to "True". If it is an optional parameter, it is set to "False".
Required	If the parameter is an input parameter this is set to "True". If it is an output parameter, it is set to "False".
Default Value	The default value, if defined, for the parameter.
Type	The type of input required for this parameter.

- The **Exit paths** section displays the exit paths for the selected tool in the order they appear in the tool step. The columns in this section are:

Column	Description
Label	The label of the exit path as it appears on the tool step.
Index	Ordinal position of the exit path.
Return Value	Exit path taken by IP at runtime.

Transfer Email Object

This tool was deprecated in CIC 4.0.

TTS Speech Modes

You can specify speech modes using the following TTS tools:

- [Play String Extended](#)
- [Play Text File Extended](#)
- [Record String Extended](#)
- [Record Text File Extended](#)

Note: NTT Hypervoice users should refer to the NTT Hypervoice documentation for a list of speech modes.

The following speech modes are available with domestic CIC's Centigram TruVoice TTS:

Identifier	Mode
{1B6BF820-9299-101B-8A19-265D428C6000}	Centigram Communications Corp. - Peter
{1B6BF820-9299-101B-8A19-265D428C6001}	Centigram Communications Corp. - Sidney
{1B6BF820-9299-101B-8A19-265D428C6002}	Centigram Communications Corp. - Eager Eddie
{1B6BF820-9299-101B-8A19-265D428C6003}	Centigram Communications Corp. - Deep Douglas
{1B6BF820-9299-101B-8A19-265D428C6004}	Centigram Communications Corp. - Biff
{1B6BF820-9299-101B-8A19-265D428C6005}	Centigram Communications Corp. - Grandpa Amos
{1B6BF820-9299-101B-8A19-265D428C6006}	Centigram Communications Corp. - Melvin
{1B6BF820-9299-101B-8A19-265D428C6007}	Centigram Communications Corp. - Alex
{1B6BF820-9299-101B-8A19-265D428C6008}	Centigram Communications Corp. - Wanda
{1B6BF820-9299-101B-8A19-265D428C6009}	Centigram Communications Corp. - Julia

The following speech mode IDs are available to international users using the L&H TTS3000:

Identifier	Mode
{534DE780-A7C7-11D1-B17B-0020AFED142E}	Lernout & Hauspie - Adult Female #1 (Michelle) American English (L&H V6.13)
{534DE781-A7C7-11D1-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #1 (Michael) American English (L&H V6.13)
{83754520-2D50-11D2-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #2 (Peter) American English (L&H V6.13) Note: Sounds like Centigram's Peter
{83754521-2D50-11D2-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #3 (Sidney) American English (L&H V6.13) Note: Sounds like Centigram's Sidney
{83754522-2D50-11D2-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #4 (Eager Eddie) American English (L&H V6.13) Note: Sounds like Centigram's Eager Eddie
{83754523-2D50-11D2-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #5 (Deep Douglas) American English (L&H V6.13) Note: Sounds like Centigram's Deep Douglas
{83754524-2D50-11D2-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #6 (Biff) American English (L&H V6.13) Note: Sounds like Centigram's Biff
{83754525-2D50-11D2-B17B-0020AFED142E}	Lernout & Hauspie - Elderly Male #7 (Grandpa Amos) American English (L&H V6.13) Note: Sounds like Centigram's Grandpa Amos
{83754526-2D50-11D2-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #8 (Melvin) American English (L&H V6.13) Note: Sounds like Centigram's Melvin
{83754527-2D50-11D2-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #9 (Alex) American English (L&H V6.13) Note: Sounds like Centigram's Alex
{83754528-2D50-11D2-B17B-0020AFED142E}	Lernout & Hauspie - Adult Female #2 (Wanda) American English (L&H V6.13) Note: Sounds like Centigram's Wanda
{83754529-2D50-11D2-B17B-0020AFED142E}	Lernout & Hauspie - Adult Female #3 (Julia) American English (L&H V6.13) Note: Sounds like Centigram's Julie
{227A0E40-A92A-11D1-B17B-0020AFED142E}	Lernout & Hauspie - Adult Female #1 (Carol) British English (L&H V6.10)
{227A0E41-A92A-11D1-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #1 (Peter) British English (L&H V6.10)
{A0DDCA40-A92C-11D1-B17B-0020AFED142E}	Lernout & Hauspie - Adult Female #1 (Linda) Dutch (L&H V6.13)
{A0DDCA41-A92C-11D1-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #1 (Alexander) Dutch (L&H V6.13)

{0879A4E0-A92C-11D1-B17B-0020AFED142E}	Lernout & Hauspie - Adult Female #1 (Véronique) French (L&H V6.13)
{0879A4E1-A92C-11D1-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #1 (Pierre) French (L&H V6.13)
{3A1FB760-A92B-11D1-B17B-0020AFED142E}	Lernout & Hauspie - Adult Female #1 (Anna) German (L&H V6.10)
{3A1FB761-A92B-11D1-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #1 (Stefan) German (L&H V6.10)
{7EF71700-A92D-11D1-B17B-0020AFED142E}	Lernout & Hauspie - Adult Female #1 (Barbara) Italian (L&H V6.11)
{7EF71701-A92D-11D1-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #1 (Stefano) Italian (L&H V6.11)
{A778E060-A936-11D1-B17B-0020AFED142E}	Lernout & Hauspie - Adult Female #1 (Naoko) Japanese (L&H V6.10)
{A778E061-A936-11D1-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #1 (Kenji) Japanese (L&H V6.10)
{2CE326E0-A935-11D1-B17B-0020AFED142E}	Lernout & Hauspie - Adult Female #1 (Carmen) Spanish (L&H V6.13)
{2CE326E1-A935-11D1-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #1 (Julio) Spanish (L&H V6.13)
{8C07B9A0-636C-11D2-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #1 (Pedro) Mexican Spanish (L&H V6.03)
{8C07B9A1-636C-11D2-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #2 (Jorge) Mexican Spanish (L&H V6.03)
{8C07B9A2-636C-11D2-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #3 (Ricardo) Mexican Spanish (L&H V6.03)
{8C07B9A3-636C-11D2-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #4 (Paco) Mexican Spanish (L&H V6.03)
{8C07B9A4-636C-11D2-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #5 (Luis) Mexican Spanish (L&H V6.03)
{8C07B9A5-636C-11D2-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #6 (Ezequiel) Mexican Spanish (L&H V6.03)
{8C07B9A6-636C-11D2-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #7 (Rogelio) Mexican Spanish (L&H V6.03)
{8C07B9A7-636C-11D2-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #8 (Carlos) Mexican Spanish (L&H V6.03)
{8C07B9A8-636C-11D2-B17B-0020AFED142E}	Lernout & Hauspie - Adult Female #1 (Josefa) Mexican Spanish (L&H V6.03)
{8C07B9A9-636C-11D2-B17B-0020AFED142E}	Lernout & Hauspie - Adult Female #2 (Isabel) Mexican Spanish (L&H V6.03)
{12E0B720-A936-11D1-B17B-0020AFED142E}	Lernout & Hauspie - Adult Female #1 (Shin-Ah) Korean (L&H V6.11)
{12E0B721-A936-11D1-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #1 (Jun-Ho) Korean (L&H V6.11)

{12E0B724-A936-11D1-B17B-0020AFED142E}	Lernout & Hauspie - Adult Female #2 (Shin-Ah 8kHz) Korean (L&H V6.11)
{06377F80-D48E-11D1-B17B-0020AFED142E}	Lernout & Hauspie - Adult Female #1 (Svetlana) Russian (L&H V6.03)
{06377F81-D48E-11D1-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #1 (Boris) Russian (L&H V6.03)
{64695500-5DC9-11D2-B17B-0020AFED142E}	Lernout & Hauspie - Adult Female #1 (Shumei) Mandarin Chinese (L&H V6.12)
{64695501-5DC9-11D2-B17B-0020AFED142E}	Lernout & Hauspie - Adult Male #1 (Dawei) Mandarin Chinese (L&H V6.12)

Update Email Object Response

This tool was deprecated in CIC 4.0.

User Queue Statistics Monitor Initiator

Gathers information about a User queue.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

User Queue

Object ID

{all}

Notification Event

Select Statistics.

Outputs

Queue Type

User.

Queue ID

The identifier for the User queue.

Interval Length

Length in seconds of the interval that this data represents.

Interval End Time

Time that this reporting interval ended.

Max Wait Time

Maximum time that calls waited in queue.

Alert Time

Total alert time of all calls alerting in queue.

Active Time

Total active time of all calls in queue.

Inactive Time

Total inactive time of all calls in queue.

Added Count

Number of calls added to the queue during this interval.

Current Alerting Count

Current number of calls alerting at time of this output.

Current Active Count

Current calls active with a user at time of this output.

Max Alerting Count

Maximum number of calls that were alerting at any point in time during this interval.

Max Active Count

Maximum number of calls that were active at any point in time during this interval.

Service Level Threshold List Size

Number of list elements in the Threshold Information Array data that follows.

Service Level Threshold

Array of threshold ids. These are the multiples of the Threshold levels set for the queue statistics. 10, 20, 30 etc.

SL Alert to First Client Connect

Number of alerting calls that became connected within Threshold Id seconds.

SL Alert to Inactive

The number of alerting calls that became inactive (abandoned).

SL Alert to Removed

The number of alerting calls removed from the queue.

SL Active to Inactive

The number of active calls that became inactive.

SL Active to Removed

The number of active calls that became inactive (hang-up).

Average Wait Time

The average wait time for alerting calls.

Exit Paths

Start

This step always exits through the start path.

Using MRCP Tools

Handler developers who want to take advantage of Media Resource Control Protocol (MRCP) can use the existing TTS tools, Play String, Play Text File, Record String, Record Text File, and the extended versions of each of those tools.

Non-Extended Tools

Each of the non-extended TTS tools automatically takes advantage of MRCP if Interaction Administrator indicates the default TTS provider is MRCP. The tools use the default language set for a server and use it to select a compatible TTS voice on the MRCP server to synthesize its text. If no compatible voice is found for a language, most MRCP servers use a default voice.

Extended Tools

The extended versions of the TTS tools can be used to take advantage of more advanced MRCP settings. You can use these tools to control common features such as voice name, volume, and rate. To specify more advanced properties, you can use name:value pairs in the tool step's **Optional Parameters** field.

The following table lists the name:value properties. When specifying a name:value pair, separate the name and value with a colon (:), for example "property1:value".

Name	Values	Description
MRCP	None	Instructs the server to use MRCP for this particular play. IMPORTANT: This parameter is required for extended versions of TTS tools. It needs to be the first parameter specified in the optional parameter field.
mrcp.audio.voice.gender	The gender of the voice when configuring MRCP TS servers in Interaction Administrator.	Enter one of the following values: <ul style="list-style-type: none">• Neutral• Male• Female

mrCP.audio.server.name	MRCP server name as specified in PureConnect.	Uses the specified MRCP server to synthesize the text to speech.
mrCP.audio.voice.name	MRCP voice name as specified in PureConnect.	Uses the specified MRCP voice to synthesize the text to speech.
*mrCP.audio.voice.rate	SSML prosody rate of a voice. The value can be an absolute float multiplier (i.e., 2.0 means twice the rate or 0.5 means half the rate), or the following predefined values which indicate a relative change: x-slow slow medium fast x-fast default	
*mrCP.audio.voice.volume	SSML prosody volume. The value can be an absolute number in the range 0-100. (Higher values are louder and zero is silence.) Or, use one of the following predefined values to indicate a relative change: silent x-soft soft medium loud x-loud default	
*mrCP.audio.language	Voice language identifiers as specified in RFC 3066 (language-country).	
mrCP.audio.vendor	The MRCP server vendor to use.	This restricts the MRCP server selection choices to ones from a particular vendor as specified in Interaction Administrator.
*mrCP.audio.contentType	The content type of the text being synthesized. Possible values: text/plain (default) application/ssml+xml text/uri-list multipart/mixed	

* These parameters are vendor-dependent and may vary between TTS engines.

Example

The following optional parameter used in the **Play String Extended** tool step specifies that MRCP is used to play SSML text.

"MRCP mrCP.audio.contentType:application/ssml+xml"

Using the SOAP Proxy Wizard

The SOAP Wizard creates a handler that exposes a SOAP interface for an existing subroutine initiator. You use the wizard to define the calling subroutine and the variables to use. The wizard builds a subroutine you can use with the SOAP tools.

To use the [SOAP wizard](#):

1. From the **Utilities** menu, point to **SOAP Wizards**, and then click **Proxy Generator**.

The **Add SOAP Proxy Wizard** appears.

2. Click **Next**.

A list of subroutine handlers appears.

3. Select the subroutine handler that the proxy calls, and then click **Next**.

Not all the subroutines handlers listed can be used. The subroutine must contain only variables of String or List of String type. If you want to use a variable of another type, you will need to create a wrapper subroutine that changes the variable type.

If the subroutine has pass-by-reference parameters, the **Subroutine Options** dialog appears.

4. Select the subroutine options, if applicable, and then click **Next**.

The **Web Service Description** page appears.

5. Enter the web service name, IIS server host name, and port.
6. Click **Next**. The **ISAPI Listener Files** page appears.
7. If the Web Services Description File and the ISAPI Filter Config File locations are correct, click **Next**.
8. Select the **Generate Failure Handling Logic** checkbox if you want to add steps to the subroutine to handle failures.
9. Select **Copy Header** if you want to copy the <header> element from the source envelope to the response envelope.
10. Click **Next**. The completion page appears.
11. Click **Finish**.

The SOAP Request subroutine is created.

Note: You still need to save, publish, and manage the handler.

See Also

[SOAP Wizard](#)

Debug command (Utilities menu)

Use this command to start a debug session. You will be prompted to open a debug handler.

Manage Handlers command (Utilities menu)

Use this command to open the Manage Handlers notebook. From this notebook you can select which published handlers are run, and which ones are not.

Utilities Stored Procedures

Opens the [DB Stored Procedure Definitions](#) window.

Variable Bar (View Menu)

The Local Variables window lists all the variables contained within the handler, the type of each variable, and the initial value of the variable.

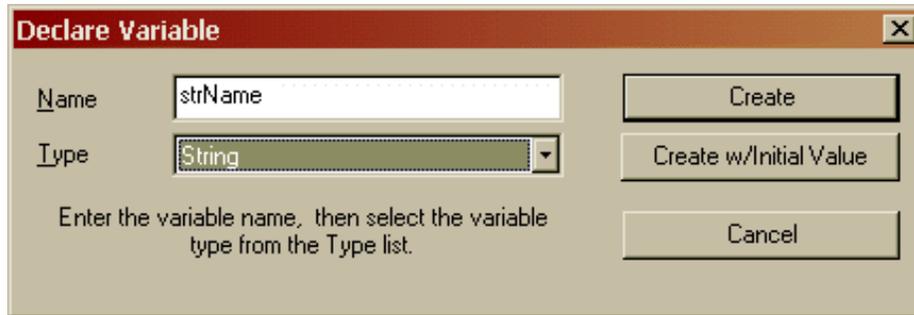
Handler authors can right-click on variables shown in the variable bar to display a menu that allows them to:

- Query what steps use the variable.
- Delete the selected variables if they are not being used in a tool step.
- Set an initial value for the variable if it is a Boolean, DateTime, Integer, Numeric or String type.

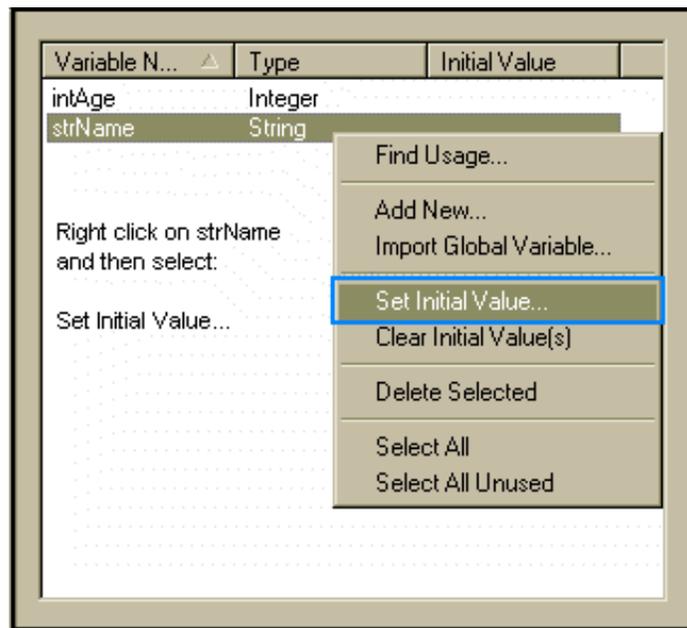
Note: Initial values are literal values only. You may not assign the initial value of a variable to an expression other than a literal expression.

- Clear the initial value
- Select all variables that are not used in tool steps.
- Add variables

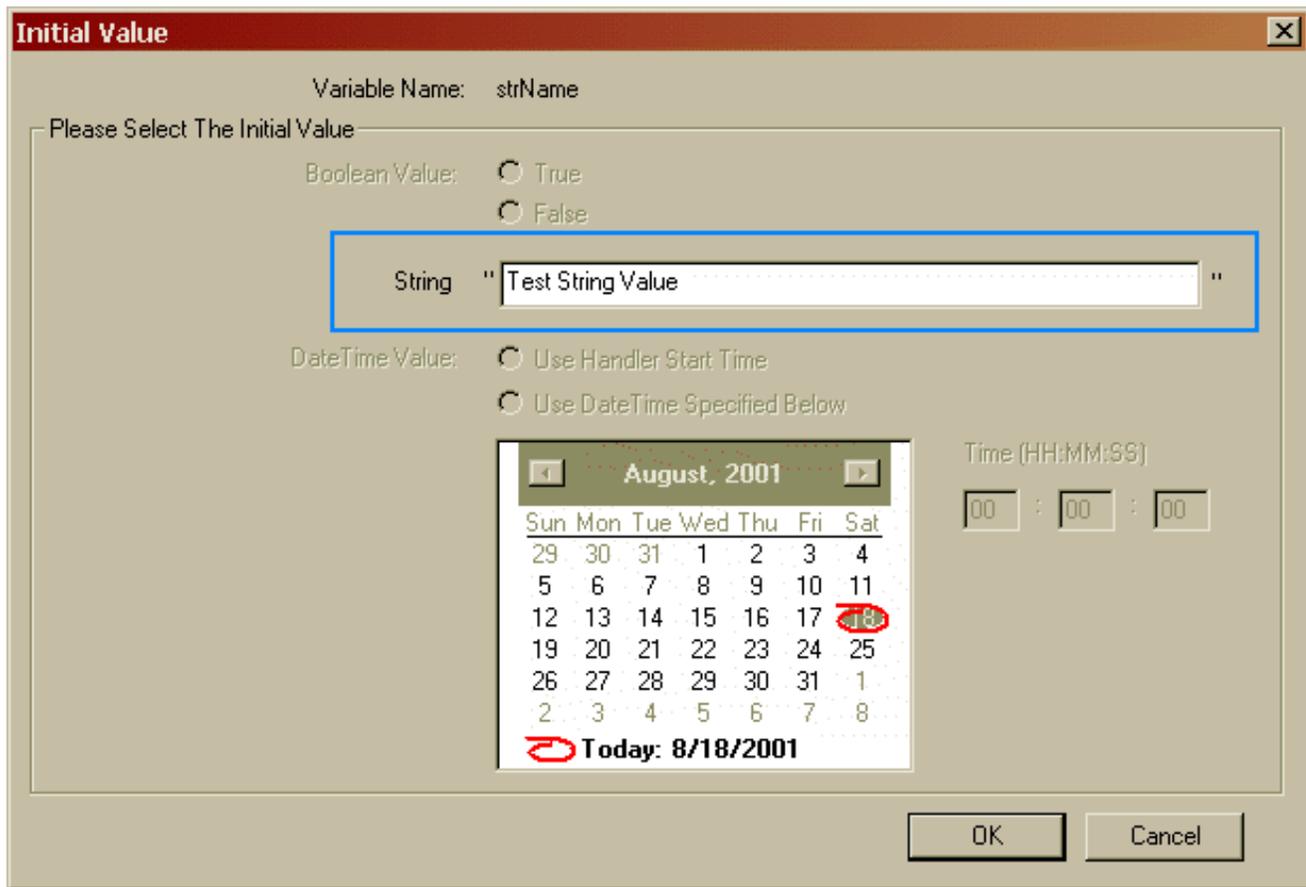
To create a new variable, select "Add New" from the menu that appears from right-clicking on the variable window. Interaction Designer will display the Declare Variable dialog. If the selected variable type for the new variable is of a type that allows for an initial value, the Create w/Initial Value button will be enabled.



At this point we can either click Create w/Initial Value or click Create and set the initial value later by right-clicking the variable in the window and selecting "Set Initial Value..." from the menu that appears. For this example, let's assume that we clicked on the Create button. Interaction Designer will automatically add the variable to the variable manager bar and then select it:



Interaction Designer now lets you assign an initial value to the variable:



For this variable we will put in "Test String Value" (without the quotes). When setting the initial values for strings you do not have to escape the string. For instance, if you had an embedded quote in the string, it is fine to enter:

Bob said "Hi"

You do **not** need to escape initial values by entering the following:

Bob said \"Hi\"

A couple of other notes about the initial value dialog:

- The dialog will enable / disable items depending on the variable type. Therefore, if you are setting the initial value for a Boolean variable, the Boolean Variable section at the top of the dialog will be enabled.
- When setting the initial value for a datetime variable, you can set the value to be either a specific date/time such as February 2, 2001 at 5:00 AM. You also have the option to set it to Use Handler Start Time which sets the value to the time when Interaction Processor launched the handler.

Deleting Unused Variables In a Handler

All unused variables in a handler can be deleted by right-clicking on the variable bar and selecting the Select All Unused menu item. The variable manager selects all unused variables. Next, right-click the variable bar and click Delete Selected from the menu. The variables are deleted from the handler.

Remember that you can only delete variables that are not being used by a tool step. The Delete Selection item is not enabled for variables that are in use by any tool. To find what tools are using a given variable, right-click that variable and select Query Usage from the menu.

Variables

Variables are containers for values. Variables exist for both normal and extended types, however you can declare variables only for normal types. Extended type variables are declared automatically when you insert a step that processes an extended type value.

Declaring Variables in a handler

There are two ways to create a variable in a handler. First, you can explicitly declare one with the [Assignment tool](#). Second, you implicitly create them when you drag a tool into a handler where that tool contains variable names. For example, if you insert an [Extended Place Call](#) step into your handler, the variable Digits is automatically declared in the handler.

Once you have declared a variable within a handler, either explicitly or implicitly, you cannot remove it.

Global Variables

All variables are scoped to the handler in which they are declared. You can pass values to a subroutine, but the variables must be declared within any handler or subroutine in which they are used.

A good way to store values outside of handlers is to store the value in a system or server parameter. You can create system and server parameters in Interaction Administrator. For example, you could create a server parameter called StrAdminEmail with the value of the system administrator's email address. A GetDSAttr step can retrieve the value of StrAdminEmail within a handler. Server parameters make handlers easy to maintain and port to other CIC servers since values are not hard coded. See [Retrieving the values of server and system parameters from handlers](#) for more information. Server parameters can store only string values, but the [type conversion operators](#) could convert them to other values types.

Related Topics

[Literal Values](#)

[Literal Values, Variables, and Operators](#)

[Operators](#)

Messages Palette

The message palette contains two tabs, a Handler Messages tab and a General Messages tab.

Handler Messages

This page lists any warnings and/or errors contained within the handler and the step number of the toolstep to which the warning or error applies. Double-clicking on the message will shift the design window's view to focus on the relevant toolstep and select that step.

General Messages

The messages shown here describe problems pertaining to Interaction Designer, but not specifically pertaining to any currently loaded handler. These could be messages pertaining to the COM API, loading libraries, loading subroutines, etc.

Status Bar command (View menu)

Use this command to display and hide the Status Bar, which describes the action to be executed by the selected menu item or selected toolbar button, and keyboard latch state. A checkmark appears next to the menu item when the Status Bar is displayed.

See [Status Bar](#) for help on using the status bar.

Toolbar command (View menu)

Use this command to display and hide the Toolbar, which includes buttons for some of the most common commands in Interaction Designer, such as File Open. A check mark appears next to the menu item when the Toolbar is displayed.

See [Toolbar](#) for help on using the toolbar.

Watch Variables

When execution is paused on a step, users can click on the Watch Variables tab in the Debug Palette and Designer will show the variable values for all variables in the currently selected for watching. For more information on watching variables in a debug session, see [View the value of a variable in a debug handler](#).

Related Topics

[Debug Palette](#)

[Step Variables](#)

[Handler Variables](#)

[Call Stack](#)

Handler Changes in CIC 4.0

This topic lists handlers that were added, changed, or deprecated since CIC 3.0 SU16 was released.

Updated Handlers

AcAvailableAgent

AcAvailableInteraction

AcInitiateGenericObject

AcProcessCategories

ACDProcessEventCall

AcProcessEventEmail

AcProcessEventGenericObject

AcProcessEventInteraction

AcProcessInQueueTimeout

AcQueueItemTimeout

B_PlayResultsAndSelect

B_PlayResultsAndSelectOrder

CiscoIPXML

ConvertEmailRecipientsToStrings

CreateDefaultAttendantMenuInit

CSSurvey_RunSurvey

CSSurveyDeleteComment

CSSurveyDisconnect

CSSurveyEntryPoint

CSSurveyGetKeys

CSSurveyPlayComment

CSSurveyPostResults

CSSurveyRecordAnswer

CSSurveyRecordComment
CustomHoldMusic
DialByName
DialByName_Audio
DialByNamePromptFilter
DialPlanEx
EmailAdminEventMsg
EMSSystemIVRCustomizations
EMSSystemIVRUserQueue
ExtendedGetDTMF
FindSIPStationByAddress
FollowMeGetUserConfig
GetBillionPrompts_lang
GetCurrencyPrompt_lang
GetDatePrompts
GetDatePrompts_lang
GetDatePromptsTZ
GetEmailHTMLBody
GetEmailInlineAttachments
GetEntityNamePrompts
GetFaxAttachmentFiles
GetIntegerPrompts_lang
GetIWebScheduleAction
GetMailboxQuotaStatus
GetMajorLanguage_lang
GetManagedServerParameter
GetMilitaryTimePrompts_lang
GetMillionPrompts_lang
GetOperatorTarget
GetPotentialUsers
GetQueueAnnouncement
GetThreeDigitsInteger_lang
GetTimePrompts_lang
GetUserClassifications
GetUserRegion
GetUserTimeZone
GetValidUsers
GetVoicemailHTMLBody
I3DialPlan
I3MessageButton
IntAttAcidAudio
IntAttAcidAudio_Audio
IntAttAdvancedStatistics

IntAttAgentTransfer
IntAttAttendantTransfer
IntAttAudioPlayback
IntAttCallerDataEntry
IntAttDBInsert
IntAttDialerCompleted
IntAttDirectProcessing
IntAttEmailAgentTransfer
IntAttEmailBuildReplay
IntAttEmailGetProfileKeyPath
IntAttEmailProcessProfile
IntAttEmailSelectionNode
IntAttEmailWorkgroupTransfer
IntAttEnhancedAudioPlayback
IntAttExpressionEvaluation
IntAttExtensionDialing
IntAttExternalTransfer
IntAttFaxBack
IntAttGetAudioPath
IntAttGetAudioSegments
IntAttGetProfileKeyPath
IntAttIncomingCall
IntAttIncomingEmail
IntAttLogging
IntAttManageAudioFiles
IntAttManageOptions
IntAttManageUnplannedSchedules
IntAttMSCRMFetchCaseByCaseID
IntAttMSRCM_PlayCases
IntAttMSRCM_PlayOrders
IntAttMSRCM_QueryCases
IntAttMSRCM_QueryCasesNoPlay
IntAttMSRCM_QueryContact
IntAttMSRCM_QueryOrders
IntAttParseEicSipHeader
IntAttProcessAcdWait
IntAttProcessError
IntAttProcessEvent
IntAddProcessEventEmail
IntAttProcessMenu
IntAttProcessMenuEmail
IntAttProcessProfile
IntAttProcessRelocation

IntAttReceiveFax
IntAttRecordPrompt
IntAttRemoteDataQuery
IntAttRemoteDataQueryProcess
IntAttRequestCallback
IntAttScreenPop
IntAttSelectionNode
IntAttSelectionNodeRHS
IntAttStationOptions
IntAttSubroutineInitiator
IntAttVoicemailTransfer
IntAttWorkgroupTransfer
InteractionAttendantEmailEntryPoint
InteractionAttendantEntryPoint
MobileOfficeDID
OnHoldAudioRandomizationMonitor
OperatorIncomingCall
OperatorProcessProfile
OutboundIncomingCall
OutboundProcessProfile
PlayCommonName
PlayDateEx_lang
PlayDateTimeTZ
PlayEntityName
PlayEnvelopeInformation
PlayEnvelopeInformation_Audio
PlayIntegerEX_lang
PlayMessageCountsEx_Audio
PlayMessageFrom
PlayTone
PlayToneAudioFile
PlayUserStatus
Prompt_ASR
Prompt_Attendant
Prompt_Hotfix
Prompt_IVR
Prompt_System
Prompt_TimeZone
Prompt_TUIMain
Prompt_TUIMessageManagement
Prompt_TUIPrivateCompanyDirectory
RecoEntryPoint
RecoGenerateCompanyDirectory

RecoGenerateCompanyDirectoryTimer
RecoGenerateDTMFGrammarString
RecoGenerateVoiceGrammarString
RecoGetInput
RecoGetResult
RecoLoadMainMenuGrammar
RecoRegisterSystemNamePrompts
RegionalDialPlanEx
ResolveVMRecipients
SetIWebStrings
SetScreenPop
Strings_Attendant
Strings_System
Strings_Web
System_AynchronousDigitsReceived
System_CallOfferingAttendantRestart
System_CallOfferingNonSystemQueue
SystemCallOfferingOutboundAttendant
System_ClientDisconnect
System_ClientPromptRequest
System_EmailOfferingInteractionAttendant
System_FaxSendCompleted
System_GenericObjectOfferingNonSystemQueue
System_HeldInteractionTimer
System_IncomingCall
System_IncomingFax
System_IncomingInteraction
System_IncomingQueueEmail
System_InitiateCallRequest
System_InteractionOfferingNonSystemQueue
System_InteractionVoicemail
System_OutgoingFax
System_OutgoingQueueEmail
System_QueueEmailOfferingNonSystemQueue
System_SendClassificationAlert
System_StationOffHook
System_SwitchhookFlash
System_TransferCallRequest
System_TransferToVoiceMailRequest
System_TUIMenu
System_VoxFormRequest
System_WebSearch
System_WorkItemObjectOfferingNonSystemQueue

System_WrapUpRequest
SystemAcdCallHolding
SystemAcdCallHolding_Audio
SystemAcdInteractionHolding
SystemAutoAnswerPage
SystemCallbackDisconnect
SystemChangeMailFolder
SystemClientPhoneticSpellings
SystemCompileNameAudio
SystemContinuousListen
SystemDescribePhoneNumber
SystemDescribeRegionalPhoneNumber
SystemDNISRouting
SystemEmailDisconnect
SystemExcelRunUsageReports
SystemFindUser
SystemGenerateUserNamePrompts
SystemGenerateUserNamePromptsProcess
SystemGenericObjectDisconnect
SystemGetOutboundANI
SystemIncomingCallAnalysis
SystemIncomingSIP
SystemIncomingTAPI
SystemInteractionDisconnect
SystemIntercomAttributeSync
SystemIVR
SystemIVRAGentQueueActivation
SystemIVRCallOfferingStationGroupQueue
SystemIVRCallOfferingStationQueue
SystemIVRCallOfferingUserQueue
SystemIVRCallOfferingWorkgroupQueue
SystemIVRChooseQueue
SystemIVRCompleteIntercomEscape
SystemIVRConfigureOneNumberFollowMeOptions
SystemIVRConfigureOneNumberFollowMeOptions_Audio
SystemIVRCustomizations
SystemIVREscape
SystemIVRFollowMeSequential
SystemIVRForceStationLogout
SystemIVRForwardMessage
SystemIVRForwardQueue
SystemIVRGetEmailIds
SystemIVRGetPhoneNumber

SystemIVRGetUserAttribute
SystemIVRGetVMDestinationQueues
SystemIVRVoicemailOptions
SystemIVRInternalVoiceMail
SystemIVRLookup
SystemIVRManageDeletedMessages
SystemIVRManageNotificationOptions
SystemIVRManagePlaybackOptions
SystemIVROfficeWideHunt
SystemIVROfficeWideHuntProcess
SystemIVROneNumberFollowMe
SystemIVRPersonalVoicemailOptions
SystemIVRPickupCall
SystemIVRPrivateDirectory
SystemIVRProcessMessage
SystemIVRReceiveFaxCall
SystemIVRReplyMessage
SystemIVRSendMessage
SystemIVRSetForwardNumber
SystemIVRSetPassword
SystemIVRSetPassword_Audio
SystemIVRSetUserAvailability
SystemIVRSetUserStatus
SystemIVRStationAutoConference
SystemIVRStationGroupQueueAlert
SystemIVRStatusForward
SystemIVRUserQueueAlert
SystemIVRUserQueueDND
SystemIVRUserQueueEmail
SystemIVRUserQueueInteraction
SystemIVRValidateUser
SystemIVRValidateUser_Audio
SystemIVRWorkgroupQueueAlert
SystemIVRWorkgroupQueueEmail
SystemLookupQueueId
SystemMessageButtonLogin
SystemMessageLightProcess
SystemMonitorObjectdisconnect
SystemNotificationCall
SystemNotifcationsProcessor
SystemParseSIPAddress
SystemProcessFollowMe
SystemProcessNotifications

SystemProvisionPhone
SystemRecordDisconnect
SystemSendWrapUpRequest
SystemSMSDisconnect
SystemSMSExpandAddresses
SystemUMDiversion
SystemVoicemailTranscription
SystemWorkgroupQueueInteraction
SystemZonePage
TargetRegionExternal
TargetRegionExtLookup
TargetRegionInternal
TargetRegionLine
TUIAudioMark
TUIAudioRecord
TUIAudioSend
TUICatchDefault
TUICatchInterpreter
TUIEntryPoint
TUIEventApplication
TUIEventMailbox
TUIEventMessage
TUIEventOptions
TUIEventPrompts
TUIEventSendMessage
TUIForwardMessage
TUIGenerateGrammars
TUIGetCalendarEvents
TUIGetDirectoryResult
TUIGetNotelId
TUIGetOPCDirectoryResult
TUIGetSpeechInput
TUIGetSpeechResult
TUIGetStatusData
TUIGetUserData
TUILoadGrammar
TUIMailProcessor
TUIMenuEventDispatch
TUIMenuInterpreter
TUIMenuInterpreterDTMF
TUIMenuInterpreterVoice
TUIMenuPlayAudio
TUIMenuQueueAudio

TUIMenuRoot
TUIPlacePrivateCall
TUIPromptListen
TUIPromptRecord
TUIPromptReset
TUIPromptSave
TUIResolveUserOption
TUISetICUserStatus
UpdateLanguagePackInfo_lang
VoiceMail

New Handlers

ACDGetSkills
CustomAcadStatisticsQuery
CustomCallAnalytics
CustomCreateXMLNode
CustomKeywordSpotted
CustomParseRoutingContexts
CustomPlayDateEx_lang
Custom_GetServerParameter
Custom_LogErrorMessage
GetMessageAttachmentFiles
IntAttProcessExtension
IntAttUpdateSkills
IntAttVoiceDataEntry
IntAttVoiceDataEntryGrammars
IntAttVoiceGenerateDTMFGrammars
IntAttVoiceMenuInput
IntAttVoiceMenuInterpreter
IntAttVoiceMenuProcess
IntAttVoiceMenuResult
SR_BuildEmailBody
SR_CheckAccess
SR_DataProcessor
SR_DBQuery
SR_FillValuesTable
SR_GetDsAttribute
SR_GetSiteId
SR_IACChangeNotification
SR_RolesChangeNotification
SR_RunScheduledReport

SR_SendEmailErrorNotification
SR_ServerParameterChangeNotification
SR_UpdateReportParametersTable
SR_UpdateReportsTables
SR_UpdateUserAccessTable
SR_WorkgroupsChangeNotification
Strings_AttendantSpeech
Strings_SR
SystemCreateXMLNode
SystemIVRCallAnalytics
SystemParseRoutingContexts
System_AcdStatisticsQuery
System_KeywordSpotted
System_OCREmail
TUIAudioRecordFile
TUIFileCleanUp
TUIGetDateOrTime
TUIGetFaxes
TUIGetFaxMessageIndicies
TUIGetICVoicemailOptions
TUIGetPhoneNumber
TUIICPlaybackOptions
UILoadUserObject
TUIOpenMailFolders
TUIPromptSetActiveGreeting
TUISetUserStatus
TUIUpdateMessageCountData
TUIUserValidated
WFM_HandlerServices

Deprecated Handlers

Custom_AcdAvailableAgent
CustomIncomingFax
CustomIncomingQueueEmail
CustomInteractionQueueItemTimeout
CustomIVRStationGroupQueueAlert
CustomWrapUpRequest
CustomWrapUpRequestIVR
CustomWrapUpRequestIVRDisconnect
Get Server Parameter
MSCRMCloseIncident
MSCRMCreateAccount

MSCRMCreateContact
MSCRMCreateIncident
MSCRMCreateProduct
MSCRMOpenIncident
MSCRMQueryAccount
MSCRMQueryContact
MSCRMQueryContract
MSCRMQueryIncident
MSCRMQueryOrder
MSCRMQueryProduct
MSCRMRetrieveAccount
MSCRMRetrieveContact
MSCRMRetrieveContract
MSCRMRetrieveIncident
MSCRMRetrieveProduct
MSCRMUpdateAccount
MSCRMUpdateContact
MSCRMUpdateIncident
MSCRMUpdateProduct
MSCRM_QueryCases
MSCRM_QueryCasesNoPlay
MSCRM_QueryContact
MSCRM_QueryOrders
MSCRM_QueryOrdersNoPlay
SystemIncomingTDD
SystemIntercomMessaging
SystemSendPagerMessage
System_InitiatePage
System_SupervisorAlert

WMF file formats for faxing

The [Create Fax Page List](#) and [Append Page tools](#) supports the following WMF file formats:

- WMF: The Windows Metafile (WMF) format is a vectored format that may or may not also contain a raster image. When reading a WMF file, Create Fax Page List and Append Pages converts all of the image data to a raster image.

For WMF files, you can read the following bits per pixel: 8, 24.

Workgroup Queue Statistics Monitor Initiator

Gathers information about a Workgroup queue.

Initiator Properties Page

For more information on the relationship between the parameters on the Initiator Properties page, see [Introduction to Initiators](#).

Notification Object Type

Workgroup Queue

Object ID

All

Notification Event

Select Statistics.

Outputs

Queue Type

Workgroup.

Queue ID

The identifier for the Workgroup queue.

Interval Length

Length in seconds of the interval that this data represents.

Interval End Time

Time that this reporting interval ended.

Max Wait Time

Maximum time that calls waited in queue.

Alert Time

Total alert time of all calls alerting in queue.

Active Time

Total active time of all calls in queue.

Inactive Time

Total inactive time of all calls in queue.

Added Count

Number of calls added to the queue during this interval.

Current Alerting Count

Current number of calls alerting at time of this output.

Current Active Count

Current calls active with a user at time of this output.

Max Alerting Count

Maximum number of calls that were alerting at any point in time during this interval.

Max Active Count

Maximum number of calls that were active at any point in time during this interval.

Service Level Threshold List Size

Number of list elements in the Threshold Information Array data that follows.

Service Level Threshold

Array of threshold ids. These are the multiples of the Threshold levels set for the queue statistics. 10, 20, 30, etc.

SL Alert to First Connected

The number of alerting calls that became connected within Threshold Id seconds.

SL Alert to Inactive

The number of alerting calls that became inactive (abandoned).

SL Alert to Removed

The number of alerting calls removed from the queue.

SL Active to Inactive

The number of active calls that became inactive.

SL Active to Removed

The number of active calls that became inactive (hang-ups).

Average Wait Time

The average wait time for alerting calls.

Exit Paths

Start

This step always exits through the start path.

WPG files formats for faxing

The [Create Fax Page List](#) and [Append Page tools](#) supports the following WPG file formats:

- WPG: The WordPerfect (WPG) format can contain vectored or raster images. CIC supports only the raster images.

For WPG files, you can read the following bits per pixel: 1, 4, 8.

Zoom 10

Sets the magnification of the design window to 10%.

Zoom 100

Sets the magnification of the design window to 100%.

Zoom 125

Sets the magnification of the design window to 125%.

Zoom 150

Sets the magnification of the design window to 150%.

Zoom 175

Sets the magnification of the design window to 175%.

Zoom 200

Sets the magnification of the design window to 200%.

Zoom 25

Sets the magnification of the design window to 25%.

Zoom 50

Sets the magnification of the design window to 50%.

Zoom 75

Sets the magnification of the design window to 75%.

Zoom In

Increases the magnification of the design window by one level.

Zoom Out

Decreases the magnification of the design window by one level.

Zoom

Increases or decreases the magnification of the active window.

Zoom Factor

The Zoom drop-down list on the toolbar allows you to select a magnification level from 10% to 200%.



File Open Dialog Box

The following options allow you to specify which file to open:

File Name

Type or select the filename you want to open. This box lists files with the extension you select in the List Files of Type box. Multiple files may be selected.

List Files of Type

Handler with the extension .ihd can be opened in Interaction Designer.

Drives

Select the drive in which Interaction Designer stores the file that you want to open.

Directories

Select the directory in which Interaction Designer stores the file that you want to open.

Network...

Choose this button to connect to a network location, assigning it a new drive letter.

Specifying Report Parameters

This topic applies to the following Report tools:

[Report Email](#)

[Report Export File](#)

[Report HTML Export](#)

[Report Print](#)

Determining Which Parameters are Used in a Report

The input values from the Report tool must correspond to the Parameter Data associated with the report. Follow these steps to determine which parameters are used in a report.

1. Open the Reports container in Interaction Administrator. Here you will find a list of all the reports that are defined for CIC.
2. Double-click on one of the reports to open the Report Configuration notebook for that report.
3. In the notebook, click the Tables/Parameter page. You will see a window that contains parameter data. This parameter data is the information needed by the report.
4. Select one of the Parameter Data entries, and then click **Edit**. The Parameter Definition dialog box appears, listing the name and value type for the selected entry.
5. If the value type is string, you need to include that name in the List of String value for the tool's String Parameter Name parameter. If the value type is Date or Time, you should include that name in the List of String value for the tool's DateTime Parameter Name parameter. If the value is Number, you should include that name in the List of String value for the tool's Number Parameter Name parameter.

Notes: The DateTime and Number parameters were used for CIC reports in IC version 2.2 and previous. In newer reports, datetime and numeric values are specified as string parameters.

When you use a report toolstep to export a historical report, use only user-supplied parameters. You can set values of other types of parameters (such as fixed to default and system-supplied) in Interaction Administrator rather than through a handler. For more information about parameter types, see "Parameter Definition for Reports" in the Interaction Administrator Help.

Examples of Boolean Parameter Names and Values

The following list provides examples of Boolean parameter names and values:

BooleanParamNames	BooleanParamValues
"Boolean1"	true
"Boolean2"	false

Examples of String Parameter Names and Values

The reports expect parameter names and values to appear in a specific format. For example, a DateTime range parameter value needs to be entered as a string in the format "YYYY/MM/DD HH:DD:SS-YYYY/MM/DD HH:DD:SS."

For user reports, the string list should contain the user name.

The following list provides examples of various types of string parameter names and values in the appropriate formats:

StringParamNames	StringParamValues
"Date"	"1962/10/31"
"DateRange"	"1991/1/17-1991/2/28"
"DateList"	"1997/12/25;1998/12/25;1999/12/25"
"DateSkipRange"	"1999/1/1-1999/2/1;1999/3/1-1999/4/1"
"DateCombo"	"1999/1/1;1999/2/1-1999/3/1"
"DateTimeRange"	"1991/1/17 11:22:33-1991/2/28 11:22:33"
"DateTimeList"	"1997/12/25 11:22:33;1998/12/25 11:22:33;1999/12/25 11:22:33"
"DateTimeSkipRange"	"1999/1/1 11:22:33-1999/2/1 11:22:33;1999/3/1 11:22:33-1999/4/1 11:22:33"
"DateTimeCombo"	"1999/1/1 11:22:33;1999/2/1 11:22:33-1999/3/1 11:22:33"
"NumberRange"	"1-99"
"NumberList"	"11;22;33"
"NumberSkipRange"	"1-2;5-6"
"NumberCombo"	"1;2-3"
"String"	"This is a string."
"StringRange"	"A-Z"
"StringList"	"Fred;Mary;Joe"
"StringSkipRange"	"A-B;D-E"
"StringCombo"	"discrete;begin-end"
"Time"	"11:22:33"
"TimeRange"	"0:0:0-23:59:59"
"TimeSkipRange"	"0:0:0-11:0:0;13:0:0-23:0:0"
"TimeCombo"	"12:0:0;13:0:0-23:0:0"
"User"	"Marketing"
"UserRange"	"A-Z"
"UserList"	"Fred;Mary;Jim"
"UserSkipRange"	"A-M;X-Z"
"UserCombo"	"Marketing;A-Z"

Examples of DateTime Parameter Names and Values

Note: The DateTime parameters were used primarily for reports created prior to IC version 2.3. Newer reports use string parameters to specify dates, times, and ranges and lists of dates and times.

The following is an example of using the DateTime parameter name and value:

DateTimeParamName	DateTimeParamValue
"DateTime"	MakeDateTimeTZ(1973, 9, 19, 5, 5, 55, "")

For more information about the MakeDateTime format, see [Creating Datetime Values](#).

Examples of Number Parameter Names and Values

Note: The Number parameters were used primarily for reports created prior to IC version 2.3. Newer reports use string parameters to specify numbers, and lists and ranges of numbers.

NumberParamName	NumberParamValue
"Number"	92.3

Substitution Fields

A substitution field is an PureConnect-specific tag in an HTML template. At handler run-time, when the Generate HTML step executes, the substitution fields are replaced with the values of the expressions bound in the step properties.

When an HTML template is selected in Interaction Designer, Interaction Designer parses the template to find the tags. From the context of the tags, Interaction Designer determines whether the binding type should be String or List of String.

Tool Changes in CIC 4.0

ACD Tools

Acid Agent Login renamed to Acid Agent Log On

Acid Agent Login Remote renamed to Acid Agent Log On Remote

Acid Agent Logout renamed to Acid Agent Log Off

Alert Server Tools

All deprecated

Email Tools (new tools)

Email Interaction Create

Email Interaction Disconnect

Email Interaction Get Message

Email Interaction Hold

Email Interaction Insert Attachment

Email Interaction Park

Email Interaction Record

Email Interaction Send Message

Email Interaction Transfer

Email Interaction Update Message

Email Object Tools

All deprecated

OCR Tools (new)

Export for OCR File

Export Parser

OCR for I3Fax Files

OCR for Tiff/Pcs/Dcx File

OCR Parser

Reco Tools

Reco Create Company Directory Grammar (updated)

Report Tools

The following [Log Tools](#) in the Reports category were deprecated in CIC 4.0:

Agent Queue Statistic Interval Log

Statistics Group Interval Log

Workgroup Queue Statistics Interval Log

SDMI Tools

All deprecated

StatAlertServer Tools (new tools)

Alert Custom Handler Result

Web Interaction Tools (new tool and initiator)

Receive Text Async

Complete Async Receive Text from an Interaction initiator

Initiators

Complete Async Receive Text from an Interaction (new)

Wrapup Required (internal use only)

Using ITTS Tools

Handler developers who want to take advantage of Interactive Text to Speech (ITTS) can use the existing TTS tools, Play String, Play Text File, Record String, Record Text File, and the extended versions of each of those tools.

For more information about installing, licensing, and configuring Interaction Text to Speech, see *CIC Text To Speech Engines Technical Reference* in the PureConnect Documentation Library.

Non-Extended Tools

Each of the non-extended TTS tools automatically takes advantage of ITTS if Interactive Administrator indicates the default TTS provider is media server. The tools use the default language set for a server and use it to select a compatible TTS voice on the media server to synthesize its text. If no compatible voice is found for a language, the media server uses a default voice.

Extended Tools

The extended versions of the TTS tools can be used to take advantage of more advanced ITTS settings. You can use these tools to control common features such as voice name, volume, and rate. To specify more advanced properties, you can use name:value pairs in the tool step's **Optional Parameters** field.

The following table lists the name:value properties. When specifying a name:value pair, separate the name and value with a colon (:), for example "property1:value".

Name	Values	Description
i3TTS	None	Indicates that media server is used for TTS. IMPORTANT: This parameter is required for extended versions of TTS tools. It needs to be the first parameter specified in the optional parameter field.
i3tts.content.language	Identifiers as specified in RFC 3066 (language-country)	Uses the specified language to synthesize the text to speech.
i3tts.content.type	text/plain (default) application/ssml+xml	The content type of the text being synthesized.
i3tts.voice.name	The i3tts voice name.	Uses the i3tts voice to synthesize the text to speech.
i3tts.voice.rate	The value can be a number in Hz or one of the following values that indicate a relative change: x-slow slow medium fast x-fast default	SSML prosody rate of a voice
i3tts.voice.volume	The value can be +/- number in dB or one of the following values that indicate a relative change: silent x-soft soft medium loud x-loud default	SSML prosody volume
i3tts.voice.pitch	The value can be number in Hz or one of the following values that indicate a relative change: x-low low medium high x-high default	SSML prosody pitch

Change log

Date	Changes
11-November-2011	CIC 4.0 introduces the following changes: Handler changes in CIC 4.0 Tool changes in CIC 4.0
10-February-2015	The CIC product suite has a new distribution model with new naming, faster release cycles, and higher quality. CIC 4.0 SU 6 was the last release using the older model. CIC 2015 R1 is first of the new releases. CIC 2015 R1 or later can be applied to any CIC 4.0 SU. CIC 2015 R2 introduces the following changes: Telephony Tools Assemble String from Attributes (updated) Verify Interaction ID (updated) Wait for Monitor End (updated)
05-May-2015	CIC 2015 R3 introduces the following changes: ACD Tools ACD Specify Interaction Skill (updated) Process Automation Tools Create PA Data renamed to Create Data Container Get PA Data Element renamed to Get Data Element Process Automation Send Handler Results renamed to Send Process Automation Handler Result Put PA Data Element renamed to Put Data Element Remove PA Data renamed to Remove Data Container OCR Tools Overview of OCR Tools (updated) Telephony Tools DateTime Attr (updated) Play String Extended (updated) Play Text File Extended (updated) Record String Extended (updated) Record Text File Extended (updated) Set DateTime Attrs (updated) Using ITTS Tools (new) Using MRCP Tools (updated)

11-August-2015	<p>CIC 2015 R4 introduces the following changes:</p> <p>Customer Interaction Center (CIC) supports several interaction management client applications. This documentation uses the term "CIC client" to refer to these applications, which include Interaction Connect, Interaction Desktop, Interaction Client .NET Edition, and Interaction Client Web Edition. Starting with CIC 2015 R3, Interaction Desktop replaced Interaction Client .NET Edition as the primary CIC client.</p> <p>ACD Tools Overview of Process Automation Tools (Updated)</p> <p>Dialer Tools Place Dialer Call (Updated)</p> <p>OCR Tools Overview of OCR Tools (updated)</p>
03-November-2015	<p>CIC 2016 R1 introduces the following changes:</p> <p>Customer Interaction Center (CIC) supports several interaction management client applications. This documentation uses the term "CIC client" to refer to these applications, which include Interaction Connect, Interaction Desktop, Interaction Client .NET Edition, and Interaction Client Web Edition. Starting with CIC 2015 R3, Interaction Desktop replaces Interaction Client .NET Edition as the primary CIC client.</p> <p>Changed Lotus Domino/Notes references to IBM Domino/Notes.</p> <p>ACD Tools Introduction to ACD Tools (updated)</p> <p>Email Tools Introduction to E-mail Tools (updated) Email Interaction Transfer (updated)</p>
09-February-2016	<p>CIC 2016 R2 introduces the following changes:</p> <p>Telephony Tools Record Call (updated)</p> <p>Initiators Interaction Administrator Change Notification Monitor Initiator (updated)</p>
03-May-2016	<p>CIC 2016 R3 introduces the following changes:</p> <p>Web Interaction Tools Snip Recording (new)</p> <p>SOAP Tools SOAP HTTP Request Ex3 (updated)</p>
09-August-2016	<p>CIC 2016 R4 introduces the following changes:</p> <p>System Tools Query Security Policy (updated)</p> <p>SMS Tools SMS Send (updated)</p>
01-November-2016	<p>CIC 2017 R1 introduces the following changes:</p> <p>List Tools List to String (updated)</p> <p>REST Tools (new) Array Builder (new) Array Parser (new) Bearer Token Request (new) JSON Builder (new) JSON Parser (new) REST HTTP Request (new)</p> <p>SMS Tools SMS Send (updated)</p> <p>System Tools RegEx Extended (updated)</p>

07-February-2017	<p>CIC 2017 R2 introduces the following changes:</p> <p>Basic Tools Table Lookup (updated)</p> <p>Telephony Tools Parallel Make Call (updated)</p>
02-May-2017	<p>CIC 2017 R3 introduces the following changes:</p> <p>System Tools Whitepages (updated)</p> <p>Telephony Tools Record Audio (updated) Record Call (updated) Record File (updated)</p>
08-August-2017	<p>CIC 2017 R4 introduces the following changes:</p> <p>Dialer Tools Place Dialer Call (updated)</p> <p>Reco Tools Introduction to Reco tools (new)</p> <p>REST Tools Bearer Token Request (updated)</p> <p>Telephony Tools DID/DNIS Routing Ex (new)</p> <p>SOAP Tools SOAP Create RPC Response (updated)</p>
31-October-2017	<p>CIC 2018 R1 introduces the following changes:</p> <p>File I/O Tools Get File Statistics (updated)</p> <p>Initiators Email Interaction Disconnected Initiator (updated) Email Interaction Incoming Initiator (updated) Email Interaction Outgoing Initiator (updated) Email Interaction Transferred Initiator(updated) Manage Survey Prompt Initiator (updated) Timer Initiator (updated)</p> <p>Telephony Tools Secure Session Get Key (updated)</p> <p>XML Tools XML Create Document From String (updated)</p>
20-February-2018	<p>CIC 2018 R2 introduces the following changes:</p> <p>PureConnect data privacy feature Customers can prevent potentially sensitive data from appearing in trace logs. See Tools impacted by the PureConnect data privacy feature.</p> <p>Report Tools Report E-mail (updated)</p> <p>Rest Tools Array Parser (updated) JSON Parser (updated) REST HTTP Request (updated)</p>

15-May-2018	<p>CIC 2018 R3 introduces the following changes.</p> <p>Fax Tools Convert PRN to I3F (deprecated)</p> <p>Host Interface Tools Introduction to Host Interface tools (updated)</p> <p>Initiators Interaction Retry Later Attempt (added) Interaction Snooze Timed Out (added)</p> <p>Reports Logging Custom Passthrough (updated)</p> <p>Web Interaction Tools Snooze Interaction (added)</p>
21-August-2018	<p>CIC 2018 R4 introduces the following changes.</p> <p>ACD Tools ACD Initiate Processing (updated)</p> <p>Fax Tools Get Fax File (updated)</p> <p>Initiators Transfer Conversation (added) Transfer Direct Message (added)</p> <p>Rest Tools Bearer Token Request (updated)</p> <p>Telephony Tools Query Media Type (updated) Secure Session Begin (updated)</p>
13-November-2018	<p>CIC 2018 R5 introduces the following changes.</p> <p>Database Tools DB Fetch (updated) DB Query (updated) DB SQLExec (updated) DB Stored Procedure (updated)</p> <p>Fax Tools Append Document Pages (updated)</p> <p>MRCP Tools Using MRCP Tools (updated)</p> <p>MIC Tools (removed) Genesys no longer supports Messaging Interaction Center (MIC).</p> <p>REST Tools Array Builder (updated) Array Parser (updated) Bearer Token Request (updated) JSON Builder (updated) JSON Parser (updated) REST HTTP Request (updated)</p>
19-February-2019	<p>CIC 2019 R1 introduces the following changes.</p> <p>ACD Tools ACD Initiate Bullseye Processing (added)</p> <p>Reports Tools IVR Event Notify (added)</p> <p>WebSphere MQ tools MQ Connect (updated)</p>
28-February-2019	Created this change log.

28-March-2019	Updated directory in SOAP HTTP Request Ex3, REST HTTP Request and Bearer Token Request tool topics.
14-May-2019	<p>CIC 2019 R2 introduces the following changes.</p> <p>Fax Tools Append Pages (updated) Create Fax Page List (updated)</p> <p>REST Tools REST HTTP Request (updated)</p> <p>SOAP Tools SOAP HTTP Request Ex3 (updated)</p> <p>System Tools Complete External Blind Transfer (updated) Complete External Call (extended) (updated)</p> <p>Telephony Tools Extended Place Call (updated) Get Attribute (updated) Get Attributes (updated)</p>
26-September-2019	Private tool step updated to indicate that it stops an ongoing recording on the call.
02-October-2019	Updated Queue Period Report Statistics Initiator to use Queue/IVR reporting (min) field in Interaction Administrator and StatServer_SendQPSNotification server parameter.
09-October-2019	<p>Updated Email Interaction Update Message, Send Email, Send Fax Mail, and Send Voicemail toolsteps to accept a string of the email address or a string of the display name with the email address in the Sender, To Recipients, CC, BCC and Reply To inputs.</p> <p>Updated REST HTTP Request toolstep that URL must be URL encoded.</p>
10-December-2019	Updated InStr() comparison operator to indicate it is case sensitive.
13-December-2019	Updated Create Fax Page List tool and Append Pages tool to indicate 32-bit version of applications required.
14-January-2020	Updated description of "Use Putback" in the Complete External Blind Transfer tool.
06-February-2020	Added VoiceXML Async Cancel tool.
30-April-2020	<p>Added note to Reco Initialize tool to indicate Server Groups input is not used.</p> <p>Updated links from my.inin.com to new locations.</p>
23-June-2020	Updated REST HTTP Request tool - Client SSL Certificate input.
20-August-2020	Removed reference to Answering Machine Detection Type parameter from Call Analysis topic.
31-October-2021	Added System_IncomingSocialMediaConversation and System_IncomingSocialMediaDirectMessage
25-October-2023	Added note to "Overview of MQSeries Tools" about the deprecation of "MQSeries tools".