



PureConnect®

2021 R4

Generated:

04-November-2021

Content last updated:

20-June-2019

See [Change Log](#) for summary of changes.



# Interaction Designer REST API Tools

## Developer's Guide

### Abstract

This document gives examples of using CIC's REST Tools to build handlers that communicate with external RESTful APIs.

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](https://help.genesys.com/pureconnect/desktop/copyright_and_trademark_information.htm).

# Table of Contents

Table of Contents	2
Introduction to Interaction Designer REST API Tools	3
CIC REST Implementation	4
REST Handler best practices	4
Proxy awareness	4
REST HTTP Request	5
REST HTTP request example	5
Bearer Token Request	8
JSON Parser	10
JSON Parser example	10
JSON Builder	11
JSON Builder example	11
Array Parser	12
Array Parser example	12
Array Builder	13
Array Builder example	13
Change Log	14

# Introduction to Interaction Designer REST API Tools

The *Interaction Designer REST API Tools Developer's Guide* is for developers who want to use Customer Interaction Center's REST capabilities to build handlers that can communicate with external RESTful APIs. For example, these tools allow CIC to communicate directly with the Salesforce API.

RESTful APIs are an alternative to SOAP-based APIs. RESTful APIs use standard HTTP methods, such as GET, PUT, POST, and DELETE to communicate, and provide endpoints, such as `http://api.example.com/user/120` for accessing and updating data. They typically use JSON as the data format, but can use other formats.

Customer Interaction Center's REST functionality removes the need to develop middleware for communicating with RESTful APIs.

# CIC REST Implementation

The CIC REST tools:

- Support GET, POST, PUT, and DELETE. The tools do not support other methods.
- Support Oauth2. The tools do not support OAuth1.
- Provide JSON and array-parsing tools. The tools do not support other formats.

CIC provides six tools for building handlers that make requests to and consume data from RESTful APIs:

- REST HTTP Request
- Bearer Token Request
- JSON Parser
- JSON Builder
- Array Parser
- Array Builder

The *Interaction Designer REST API Tools Developer's Guide* provides examples of using these tools in handlers. For more information about the inputs, outputs, and exit paths of each tool, see the *Interaction Designer Help* at [https://help.genesys.com/cic/mergedProjects/wh\\_id/desktop/hid\\_introduction.htm](https://help.genesys.com/cic/mergedProjects/wh_id/desktop/hid_introduction.htm).

## REST Handler best practices

- Familiarize yourself with the API you want to communicate with. For example, some APIs receive parameters in the URL itself, while others take parameters in the header. This behavior has an impact on tool inputs.
- Be aware of any incompatibilities between an API and the CIC REST tools. For example, these tools do not support OAuth1.

## Proxy awareness

REST HTTP Request and Bearer Token Request include an optional Proxy Uri parameter where you can enter a forward proxy address for retrieving content from the API server. The parameter must include the protocol used. For example, `http://proxy.example.com` or `http://192.168.1.10`.

# REST HTTP Request

Use the REST HTTP Request tool to send a request to an external REST service. You can optionally use client certificates with this tool. The REST HTTP Request 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 the Windows Certificate store to check for validity of the certificate.
- Uses openssl to do the verification.

## REST HTTP request example

This example makes a request to the Twitter API. It uses the following variables:

Variable Name	Type	Initial Value
BearerToken1	String	
c_DELETEHeader	String	header1: val1header2: val2header3: val3
c_GETHeaderList	String	
c_POSTHeaders	String	
c_PUTHeader	String	
CustomDataList	List of S...	
CustomEventId	String	
CustomObjectId	String	
HTTPRawResponseBody1	String	
InitiatorEventName	String	
InitiatorObjectName	String	
InitiatorObjectTypeName	String	
IsHeaderList	List of S...	
sBearerToken1	String	
sBody	String	
sHTTPRawResponseBod...	String	
sMethodType	String	
sUri	String	https://api.twitter.com/1.1/statuses/user_timeline.json?count=10&screen_name=twitterapi
Twit_Key	String	qarW57P7LV1zfmSeECLpoKsjy
Twit_Secret	String	kXtRcry31ys7gq0LEi4zouTxx3048mC28L2hwkCoisZeZpfaFI
URL_Twit_GetToken	String	https://api.twitter.com/oauth2/token
URL_Twit_Timeline	String	https://api.twitter.com/1.1/statuses/user_timeline.json?count=10&screen_name=twitterapi

Properties of REST HTTP Request (18)

General | Inputs | Inputs | Outputs

URL: sUri

Proxy Uri: ?

HTTP Method: sMethodType

Additional HTTP Headers: ?

Bearer Token: sBearerToken1

Content Type: "application/json"

Raw Request Body: ""

Request Timeout [s]: 20.0

Client SSL Certificate: false

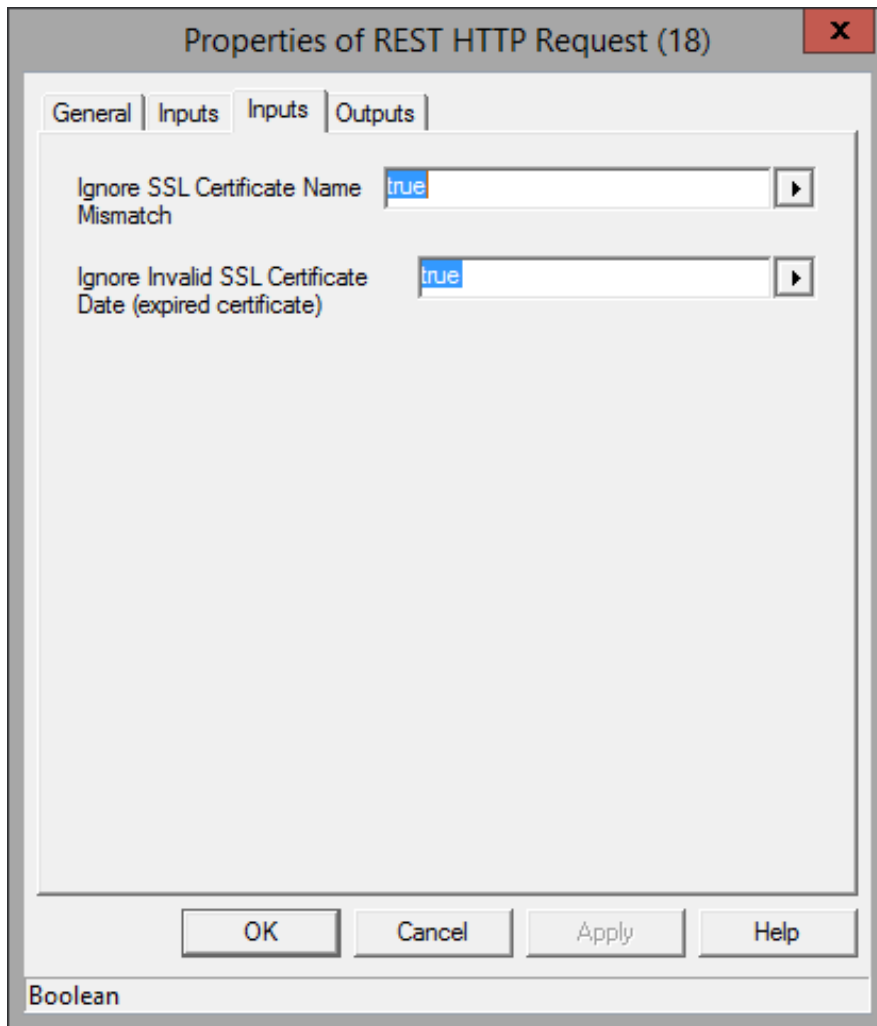
Ignore Unknown SSL Certificate Authority: true

Ignore Wrong SSL Certificate Usage: true

OK Cancel Apply Help

String

REST HTTP Request inputs



REST HTTP Request inputs continued

# Bearer Token Request

Use this tool to request an OAuth 2.0 access bearer token for a client credentials grant or a password grant. This tool returns a parsed token that you can use in the Bearer Token parameter in the REST HTTP Request tool. Like the REST HTTP Request tool, 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 the Windows Certificate store to check for validity of the certificate
- Uses openssl to do the verification.

In the following example, the handler makes a request to Twitter for a client credentials grant.

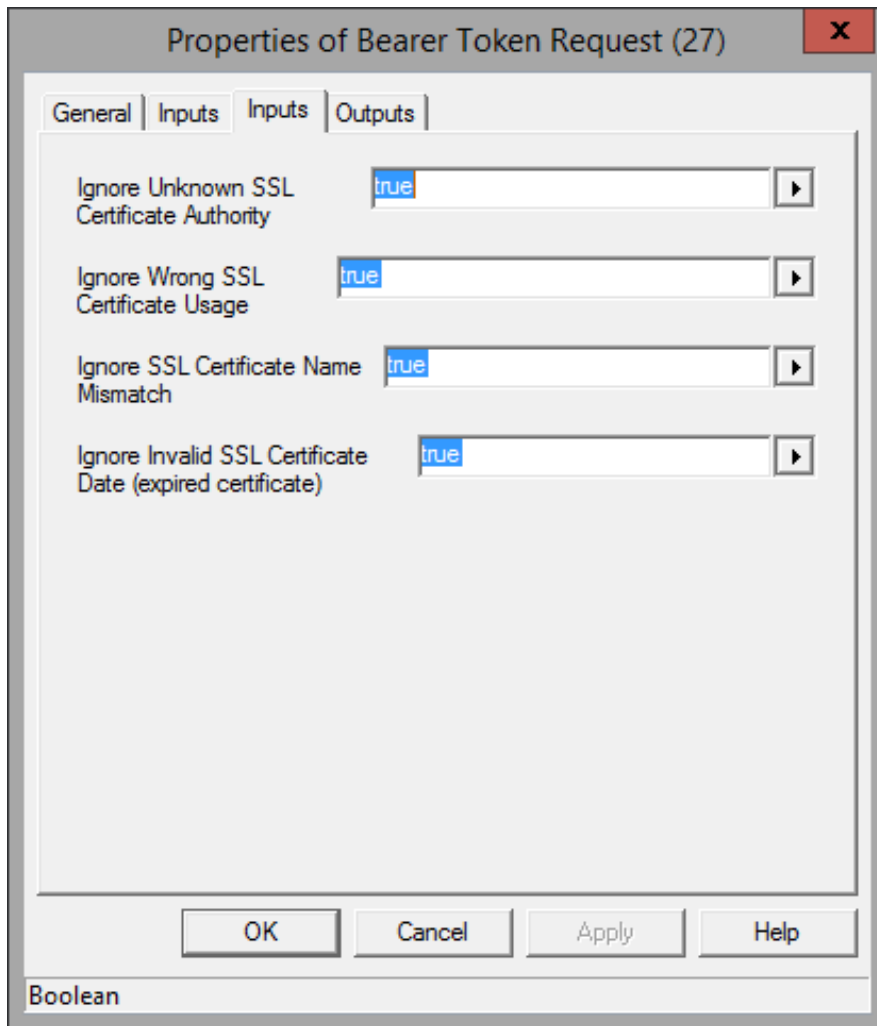
The screenshot shows a dialog box titled "Properties of Bearer Token Request (27)". It has four tabs: "General", "Inputs", "Inputs", and "Outputs". The "Inputs" tab is selected. The dialog contains the following fields and values:

Field	Value
URL	URL_Twit_GetToken
Proxy Uri	?
ID/Key	Twit_Key
Secret	Twit_Secret
User Name	?
Password	?
Grant Type	"client_credentials"
Credentials in POST Body?	false
Content Type	"application/x-www-form-urlencoded"
Request Timeout [s]	10.0
Client SSL Certificate	false

At the bottom of the dialog are four buttons: "OK", "Cancel", "Apply", and "Help". Below the dialog box, the text "String" is visible.

Bearer Token Request Inputs





Bearer Token Request Inputs continued

Variables		
Variable Name	Type	Initial Value
BearerToken1	String	
c_DELETEHeader	String	header1: val1header2: val2header3: val3
c_sGETHeaderList	String	
c_sPOSTHeaders	String	
c_sPUTHeader	String	
CustomDataList	List of S...	
CustomEventId	String	
CustomObjectId	String	
HTTPRawResponseBody1	String	
InitiatorEventName	String	
InitiatorObjectName	String	
InitiatorObjectTypeName	String	
IsHeaderList	List of S...	
sBearerToken1	String	
sBody	String	
sHTTPRawResponseBod...	String	
sMethodType	String	
sUrl	String	https://api.twitter.com/1.1/statuses/user_timeline.json?count=10&screen_name=twitterapi
Twit_Key	String	qarW57P7LV1zfmSeECLpoKsjy
Twit_Secret	String	kXtRcry31ys7gq0LEi4zouTxx3048mC28L2hwkCoisZeZpfaFI
URL_Twit_GetToken	String	https://api.twitter.com/oauth2/token
URL_Twit_Timeline	String	https://api.twitter.com/1.1/statuses/user_timeline.json?count=10&screen_name=twitterapi

Variables for Bearer Token Request example

# JSON Parser

This REST tool outputs a list of names and a list of values from a JSON object. If you have nested objects, the parser places the entire nested object in the values list as a string. To parse nested JSON objects fully, use multiple JSON Parser steps.

## Important!

- Names must be unique. If names aren't unique, JSON Parser does not throw an error, but the JSON is not correctly parsed. For example, {"user": "johndoe", "user": "janedoe"} does not cause an error, but is not parsed correctly.
- Starting with Interaction Designer 2017 R3, JSON Parser can parse a JSON object that includes these keywords: break, case, catch, class, const, continue, debugger, default, delete, do, else, enum, export, extends, finally, for, function, if, implements, import, in, instanceof, interface, let, new, package, private, protected, public, return, static, super, switch, this, throw, try, var, void, while, with, yield. Previously, a JSON object that contained a keyword would cause the JSON Parser to exit with the failure path.

## JSON Parser example

Given the following JSON object, JSON Parser returns three list of string variables: one for the names, one for the values, and one for the data types of each item.

```
{
  "user": "johndoe",
  "admin": false,
  "uid": 1000,
  "groups": ["users", "wheel", "audio",
    "video"], "innerJSON": {
  "observer": "janedoe",
  "readonly": true,
  "uid": 1001,
  "watchlists": ["Aveeno", "Purell",
    "Blistex", "Carmex"]
}
}
```

JSON string	
IsNamesList1	5 item(s) in list.
[0]	admin
[1]	groups
[2]	innerJSON
[3]	uid
[4]	user
IsValueTypeList1	5 item(s) in list.
[0]	bool
[1]	array
[2]	object
[3]	double
[4]	string
IsValuesList1	5 item(s) in list.
[0]	0
[1]	["users", "wheel", "audio", "video"]
[2]	{"observer": "janedoe", "readonly": true, "uid": 1001, "watchlists": ["Aveeno", "Purell", "Blistex", "Carmex"]}
[3]	1000
[4]	johndoe

# JSON Builder

This REST tool accepts a list of names and a list of values (and the value data types) to build a JSON object of name/value pairs.

## JSON Builder example

Given the input variables in the JSON Parser example, JSON Builder produces a JSON object:

JSON Builder	
IsNamesList1	5 item(s) in list.
[0]	admin
[1]	groups
[2]	innerJSON
[3]	uid
[4]	user
IsValueTypeList1	5 item(s) in list.
[0]	bool
[1]	array
[2]	object
[3]	double
[4]	string
IsValuesList1	5 item(s) in list.
[0]	0
[1]	["users", "wheel", "audio", "video"]
[2]	{ "observer": "janedoe", "readonly": true, "uid": 1001, "watchlists": ["Aveeno", "Purell", "Blistex", "Carmex"] }
[3]	1000
[4]	johndoe

Resulting object:

```
{
  "user": "johndoe",
  "admin": false,
  "uid": 1000,
  "groups": ["users", "wheel",
  "audio", "video"], "innerJSON": {
  "observer": "janedoe",
  "readonly": true,
  "uid": 1001,
  "watchlists": ["Aveeno",
  "Purell", "Blistex", "Carmex"]
  }
}
```

# 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.

## Array Parser example

The handler used in the array examples has the following variables:

Variable Name	Type	Initial Value
CustomDataList	List of String	
CustomEventId	String	
CustomObjectId	String	
InitiatorEventName	String	
InitiatorObjectName	String	
InitiatorObjectType	String	
JSONstring1	String	
IsNamesList1	List of String	
IsValuesList1	List of String	
IsValueTypeList1	List of String	
s_DelimitedName	String	sean stevelseth finn
s_DelimitedValue	String	blue red yellow green
s_DelimitedValue	String	string string string int
sArray	String	["green", 10, false, ["Aveeno", "Purell", "Blistex", "Carmex"]]
sArrayString1	String	

### Variables for array examples

Given the following array, Array Parser returns list of string variables for the names, values, and data types of the array items.

```
["green", 10, false, {"user":  
"johndoe", "admin": false, "uid":  
1000, "groups": ["users", "wheel",  
"audio", "video"]}, ["Aveeno", "Purell",  
"Blistex", "Carmex"]]
```

IsValueTypeList1	5 item(s) in list.
[0]	<u>string</u>
[1]	<u>double</u>
[2]	<u>bool</u>
[3]	<u>object</u>
[4]	<u>array</u>
IsValuesList1	5 item(s) in list.
[0]	<u>green</u>
[1]	<u>10</u>
[2]	<u>0</u>
[3]	<u>{"admin":false, "groups":["users", "wheel", "audio", "video"], "uid":1000, "user":"johndoe"}</u>
[4]	<u>["Aveeno", "Purell", "Blistex", "Carmex"]</u>

# 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.

## Array Builder example

Given the variables in the Array Parser example, Array Builder outputs an array:

IsValueTypeList1	5 item(s) in list.
[0]	<u>string</u>
[1]	<u>double</u>
[2]	<u>bool</u>
[3]	<u>object</u>
[4]	<u>array</u>
IsValuesList1	5 item(s) in list.
[0]	<u>green</u>
[1]	<u>10</u>
[2]	<u>0</u>
[3]	<u>{ "admin":false, "groups":["users", "wheel", "audio", "video"], "uid":1000, "user":"johndoe"}</u>
[4]	<u>["Aveeno", "Purell", "Blistex", "Carmex"]</u>

Resulting array:

```
["green", 10, false, {"user":  
"johndoe", "admin": false, "uid":  
1000, "groups": ["users", "wheel",  
"audio", "video"]}, ["Aveeno", "Purell",  
"Blistex", "Carmex"]]
```

# Change Log

The following table lists the changes to the *Interaction Designer REST API Tools Developer's Guide* since its initial release.

Date	Change
01-June-2016	Initial release of this document.
21-December-2016	Updated JSON Parser to list keywords.
01-August-2017	Updated cover page, copyright and trademark information. Rebranded this document to apply Genesys terminology.
28-March-2019	Updated REST HTTP Request and Bearer Token Request topics to correct directory I3\IC\Certificates\REST.
20-June-2019	Reorganized the content only, which included combining some topics and deleting others that just had an introductory sentence such as, "In this section...".