PureConnect®

2023 R2

Generated:

30-May-2023

Content last updated:

20-June-2019

See Change Log for summary of changes.

GENESYS™

# Interaction ScriptAssist for Oracle Service Cloud

## Technical Reference


### Abstract

This document describes Interaction ScriptAssist for Oracle Service Cloud, a tool that creates scripted screen pops used in the PureConnect integration to Oracle Service Cloud customer relationship management (CRM) services.

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

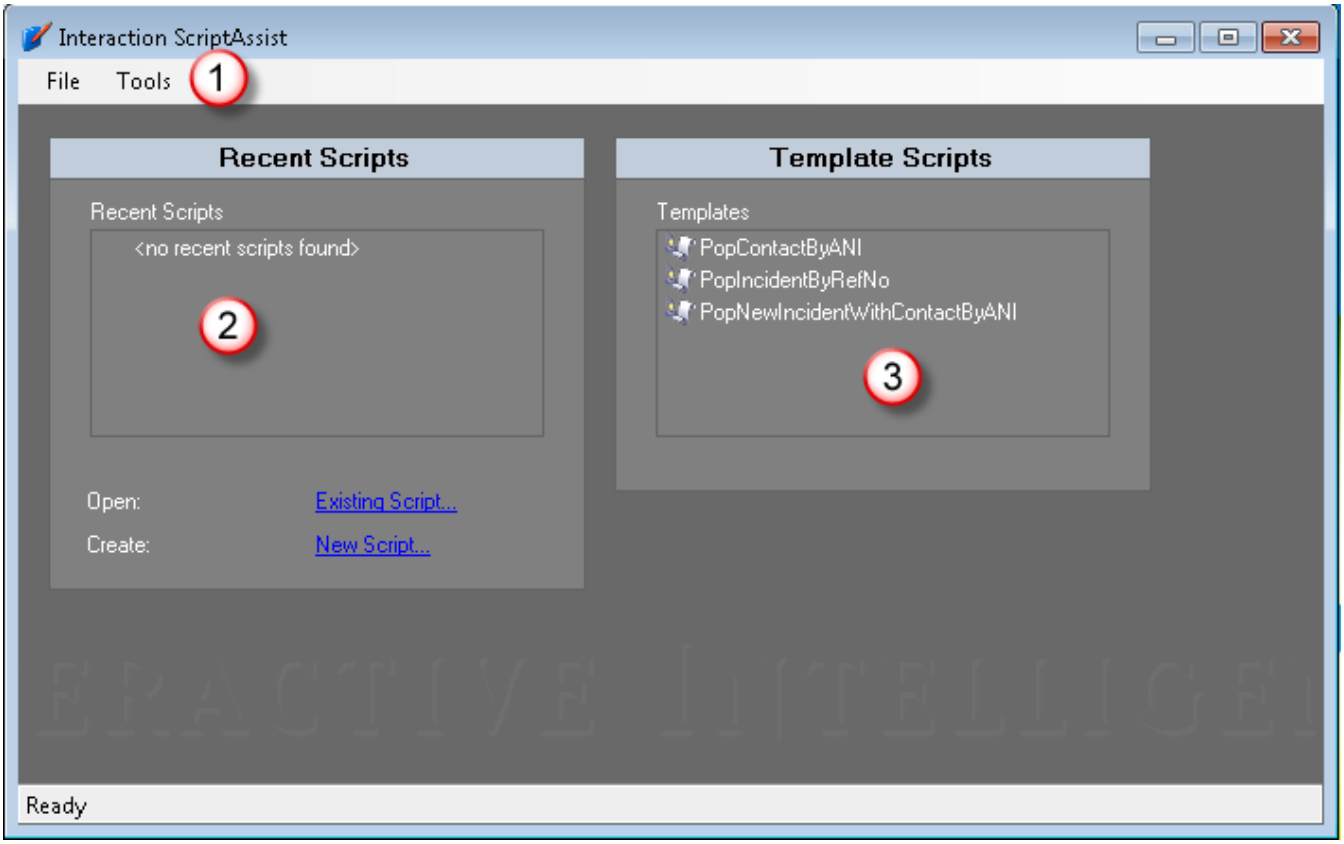
# Table of Contents

# Features of Interaction ScriptAssist for Oracle Service Cloud

PureConnect's Interaction ScriptAssist for Oracle Service Cloud (*ScriptAssist*) allows you to use the objects, properties, and events in the Oracle Service Cloud Connect Web Services for SOAP application programming interface along with the Oracle Service Cloud Desktop Add-Ins application programming interface to create screen pop scripts simply by placing objects onto a workspace and adjusting their properties and event handlers.

# ScriptAssist start page

The first time you start ScriptAssist, you see the start page.




## Basic page elements

The menu bar (1):

> On the **File** menu, you can create new scripts, open existing scripts on your server, import a script from an XML file, and do related tasks. On the **Tools** menu, you can configure your copy of ScriptAssist to work with your IC server and set other options.

The Recent Scripts section (2):

> This shows a list of recently used scripts on your IC server. In the figure above, this area is blank because when you start ScriptAssist for the first time, you have not yet created any scripts.

The Template Scripts section (3):

> This shows a list of templates used to create scripts for common tasks. The ScriptAssist package includes these scripts.

## Template scripts

ScriptAssist comes with several bundled script templates. You can use these templates as a starting point and then modify them for your own specific needs:

PopContactByANI:

> This template creates a script that searches the server for a contact record and then opens the record for editing.

PopIncidentByRefNo:

> This template creates a script that searches the server for a specific incident record and then opens the record for editing.

PopNewIncidentWithContactByANI:

This template creates a script that searches the server for a specific contact record and then creates an incident based on the record.

## Edit a script

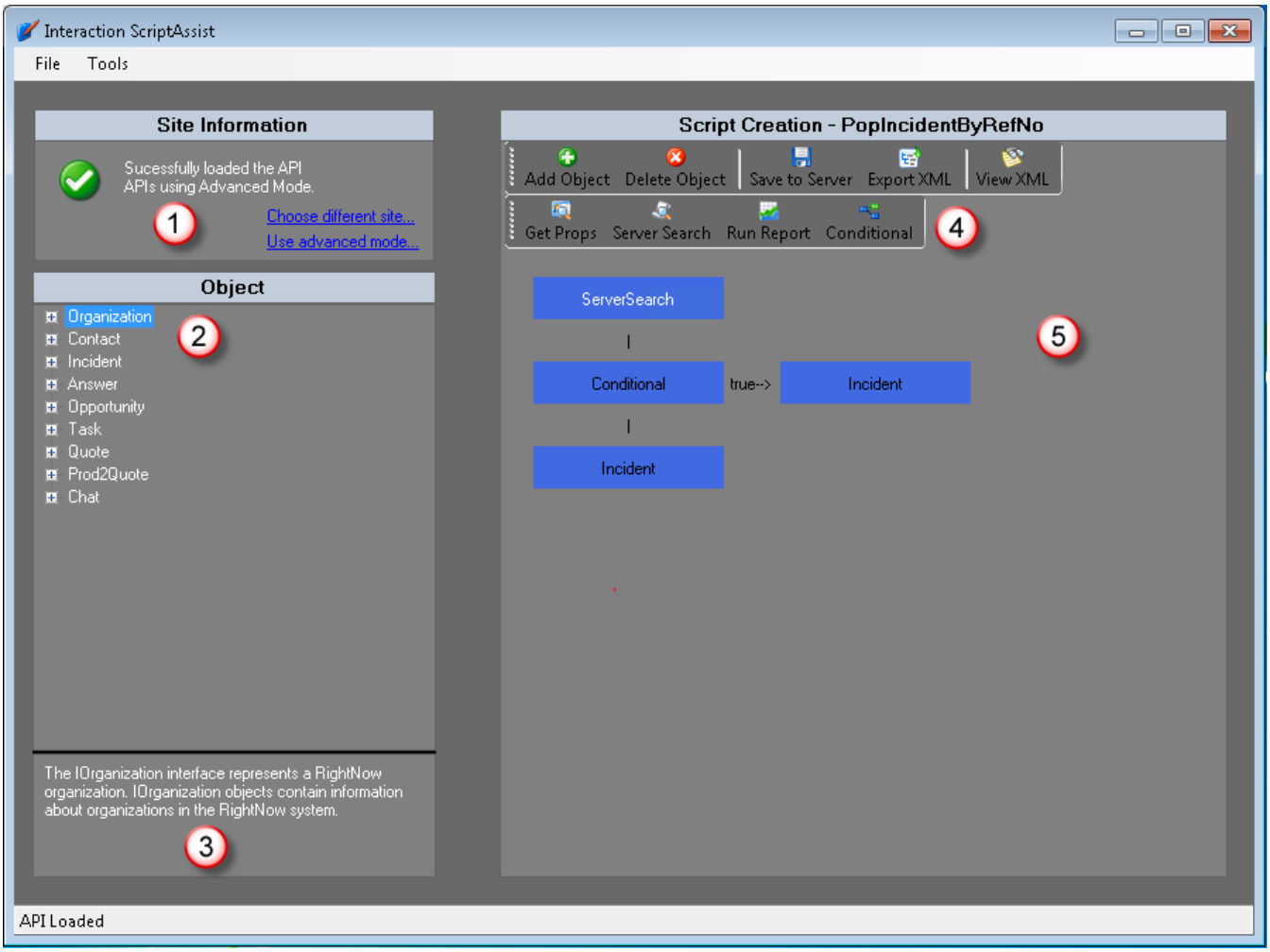
Following are the basic options for editing a script:

- On the right of the screen is a box called **Template Scripts**. To create a script starting with a template. For more information, see Use the templates.
- Below the **Recent Scripts** section is a link to open an Existing Script. Existing Scripts are XML files on the network. You will browse to the location of those scripts. For more information, see Use existing scripts.
- Below the **Recent Scripts** is a link to create a New Script. This is used when you want to create a script from scratch. For more information see Create a new script.
- Another option is to use the **File** menu and select **Open Server Script**. This will cause a connection to the server. For more information, see Open a server script.

**Note:**
When ScriptAssist starts, it checks to see if the Oracle Service Cloud client is installed on your computer. In particular, it looks in your `&percnt;appdata&percnt;\RightNow_Technologies\<name of your site>`. In this folder, the Oracle Service Cloud client installs files needed for ScriptAssist to work properly. ScriptAssist will not work unless you previously installed and set up the Oracle Service Cloud client on your computer.

# ScriptAssist editor page

Once you have chosen one of the 4 options above you will open the editor page. If you have not previously connected to a site you will get a pop-up that tells you that you must connect to a site, click **OK**. Then enter the Site Name. The Site Name is the Oracle Service Cloud interface name.




## Basic page elements

The Site Information section (1):

> This section shows the Oracle Service Cloud site you have selected. The **Choose different site** link lets you change to a different site.

The Object section (2):

> This section lists the basic node types that you can use in your script. By clicking the Plus (+) sign to the left of each type, you see a list of its properties. When you highlight a property, ScriptAssist displays its description in the description section (3). For more information, see "Basic node types (objects)" in Basic features of ScriptAssist.

The description section (3):

> This displays a description of the node type or property highlighted in the Object section (2). The information comes from Oracle Service Cloud's own API documentation.

The Script Creation toolbar (4):

> This contains buttons for script-editing operations, such as adding nodes, deleting nodes, and exporting a script to XML. This also has buttons for inserting special node types. For more information, see "Script toolbars" and "Special node types (operations)" in Basic features of ScriptAssist.

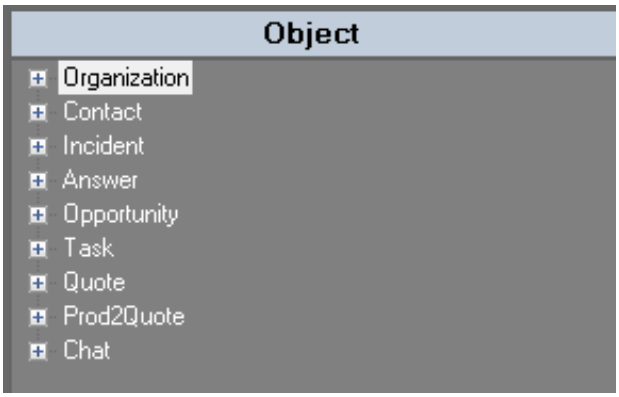
The Script Creation workspace (5):

This section displays a tree diagram of the script you are currently editing. Double-click script nodes to do further operations on them.

# Basic features of ScriptAssist

Following are the basic features of Interaction ScriptAssist for Oracle Service Cloud. For how-to information, see Basics of scripts in ScriptAssist for Oracle Service Cloud.

## Basic node types (objects)



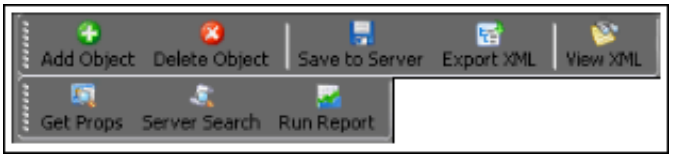

ScriptAssist lets you add script actions that use these standard Oracle Service Cloud record types:

- Organization: This is a node type that is used to create, edit, and delete organization records in your screen pop. For more information, see the IOrganization section of your Oracle Service Cloud documentation.
- Contact: This is a node type that is used to create, edit, and delete contact records in your screen pop. For more information, see the IContact section of your Oracle Service Cloud documentation.
- Incident: This is a node type that is used to create, edit, and delete incident records in your screen pop. For more information, see the IIncident section of your Oracle Service Cloud documentation.
- Answer: This is a node type that is used to create, edit, and delete answer records in your screen pop. For more information, see the IAnswer section of your Oracle Service Cloud documentation.
- Opportunity: This is a node type that is used to create, edit, and delete opportunity records in your screen pop. For more information, see the IOpportunity section of your Oracle Service Cloud documentation.
- Task: This is a node type that is used to create, edit, and delete task records in your screen pop. For more information, see the ITask section of your Oracle Service Cloud documentation.
- Quote: This is a node type that is used to create, edit and delete quote records in your screen pop. For more information, see the IQuote section of your Oracle Service Cloud documentation.
- Prod2Quote: This is a node type that is used to create, edit and delete product records that were used in quotes in your screen pop. For more information, see the IProd2Quote section of your Oracle Service Cloud documentation.
- Chat: This is a node type that is used to create, edit and delete chat records your screen pop. For more information, see the IChat section of your Oracle Service Cloud documentation.

**Note:**
For more information about these node types, see your Oracle Service Cloud Desktop Add-Ins documentation.

## Script toolbars

After the user selects a script from the start page, ScriptAssist loads the script for editing. ScriptAssist then displays the script toolbars. ScriptAssist displays these toolbars only when a script is open for editing.




From these toolbars, you can:

- Add an object node to the script after the currently selected node.
- Delete the currently selected node from the script.
- Save the current script to the IC server.

- Export the current script to an XML file for backup.
- View the current script in XML code to analyze its operation.
- Insert a **Get Properties** node after a **Saved** event to copy one or more saved properties for use later in the script.
- Insert a **Server Search** node at the current script position to search the server for specific answers, contacts, incidents, opportunities, or organizations.
- Insert a **Run Report** node to run a predefined report from the Oracle Service Cloud server.

## Special node types (operations)


ScriptAssist lets you script specially defined operations. The following types (available from the script toolbar) represent special nodes:

- Get Prop: After a **Save** action, this copies values you select and passes them to any subsequent nodes in your script. For example, the user might save a new customer record. The **GetProperties** node could then obtain the newly defined customer ID and pass that value to an **Incident** node that occurs later in the script.
- Server Search: This searches the Oracle Service Cloud site through the Connect Webservice SOAP API for records based on criteria you enter in the dialog box.
- Run Report:This causes the script to run a report defined in the Oracle Service Cloud site. Specify the ID of the report. For more information, see Use a Run Report node.
- Conditional: This is a way to set up a True/False condition. You are able to define criteria based on Equal, Not Equal, Greater Than or Less Than. Then determine what to do if the condition is True and what to do when the condition is False.

**Note:**
For more information about these Oracle Service Cloud node types, see your Oracle Service Cloud Desktop Add-Ins documentation. Also, be aware that Oracle Service Cloud refers to the **ServerSearch** node simply as "Search" and to the **Run Report** node as "RunReport." Oracle Service Cloud does not use the **GetProperties** node type.

# Basics of scripts in ScriptAssist for Oracle Service Cloud

Before you use Interaction ScriptAssist for Oracle Service Cloud, Genesys recommends that you configure it to connect both with your IC server and with the Oracle Service Cloud server.

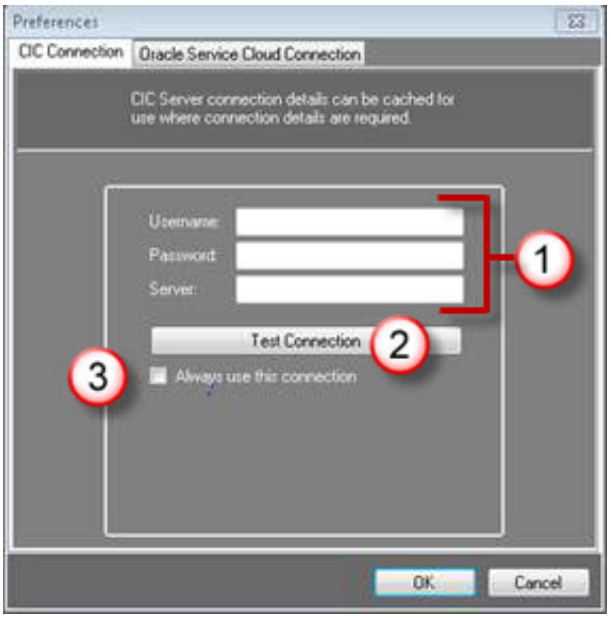


## Set preferences for Interaction ScriptAssist

Before you create or edit scripts, set preferences for ScriptAssist to work with your IC server and with the Oracle Service Cloud server. You set preferences in the **Options** dialog box, which has two tabs discussed in the following sections.

1. Click **Tools** on the menu.
2. Choose **Options**.

### Options | CIC Connection tab

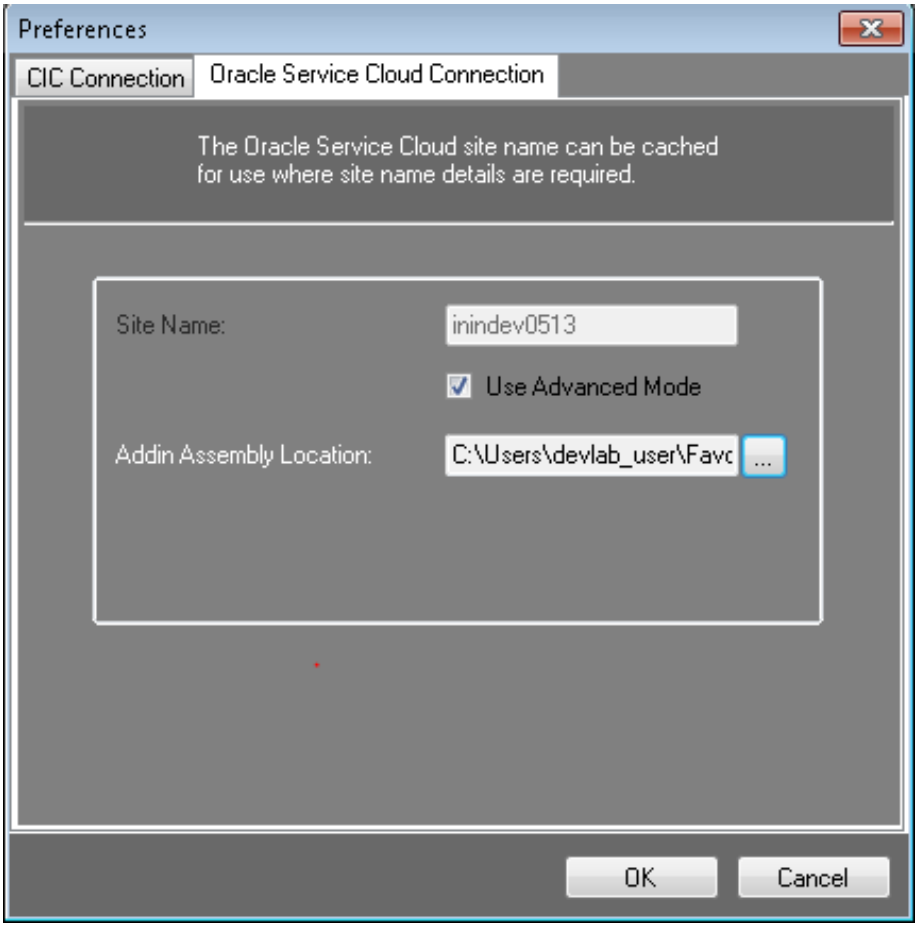



On the **CIC Connection** tab, you can:

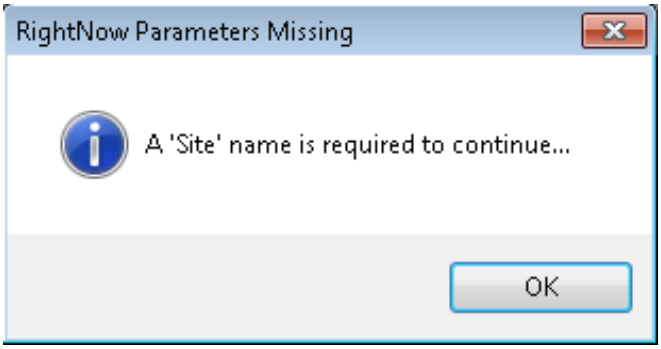
- Enter your CIC server user ID and password (1).
- Enter the name of your CIC server (1).
- Test your connection to the CIC server (2). The **Test Connection** button causes ScriptAssist to try to establish a connection with the CIC server. If the logon information is incorrect or if there are network problems, the connection fails and you can attempt to diagnose the problem.
- Tell ScriptAssist always to use the CIC server and logon information that you provided on this tab (3).

The **Always use this connection** check box makes ScriptAssist's start page show a list of scripts on the CIC server you named. Edit these scripts simply by clicking their names in the list. When you save scripts, ScriptAssist automatically populates the **Save** dialog box with your CIC server and logon credentials. If you do not select this check box, then ScriptAssist does not use the logon credentials.

### Options | Oracle Service Cloud Connection tab




On the **Oracle Service Cloud Connection** tab, you can enter the name of the site to use to load the Oracle Service Cloud Desktop Add-Ins API. If you do not enter the site name, then when Oracle Service Cloud opens a script, it will display a dialog box requesting the site name.




## Use Advanced Mode

**Note:**
The *>Advanced Mode* for the Oracle Service Cloud Connection was added to allow the ability to specify where the Addin assembly files are located. This is required for customers running the November '11 (RightNow) and later site versions of Oracle Service Cloud.

Select the **Use Advanced Mode** check box. Then, browse to where the Addin assemebly DLL's are located.

For Windows 7/Server 2008 machines, the files should be here:

`C:\Users\<user>\AppData\Roaming\RightNow_Technologies\<site>\<build number>\AddInPipeline\AddInViews\RightNow.AddIns.AddInViews.dll`

For older OS's (XP/Server 2003), the files should be here:

`C:\Documents and Settings\<user>\ApplicationData\RightNow_Technologies\<site>\<build number>\AddInPipeline\AddInViews\RightNow.AddIns.AddInViews.dll`

**Note:**
These locations could change, depending on any change in structure as new versions of Oracle Service Cloud are released.

Once the files have been specified, click **OK**.

## Basics of building scripts

1. The ScriptAssist GUI interface shows each action as an object.
2. When saving the scripts, you can save them in XML format to a network drive or a local drive. Or you can save directly to the CIC server.
3. In order to run, the script it must be saved to the CIC server.

## Use the templates

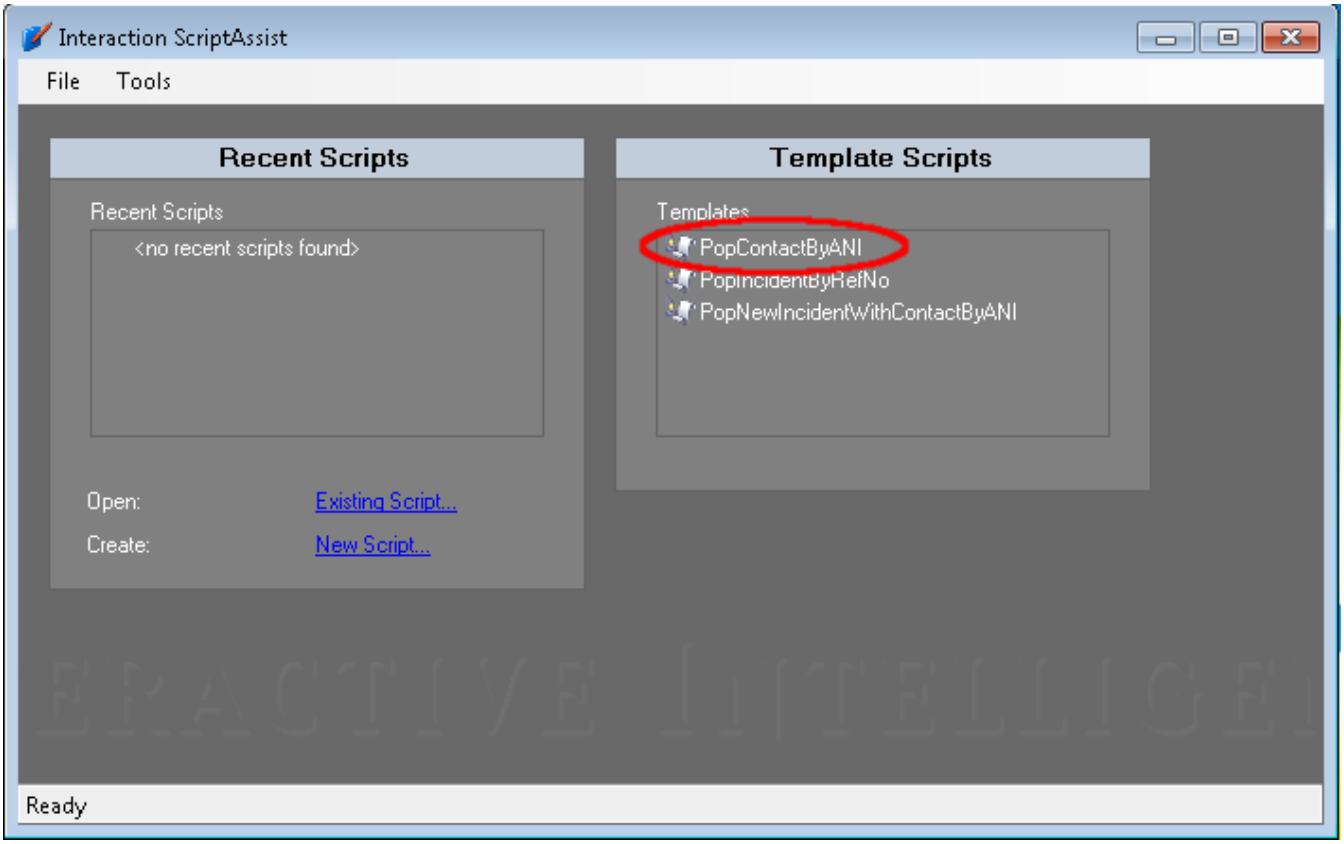
To use a template, simply click the name of the template you want to use. After making the changes to the template, click **File > Save to Server**. You can then give it a new name or use the template name. You will then also set the server to save it to as well as the user name and password.

For more information about the templates, see the following:

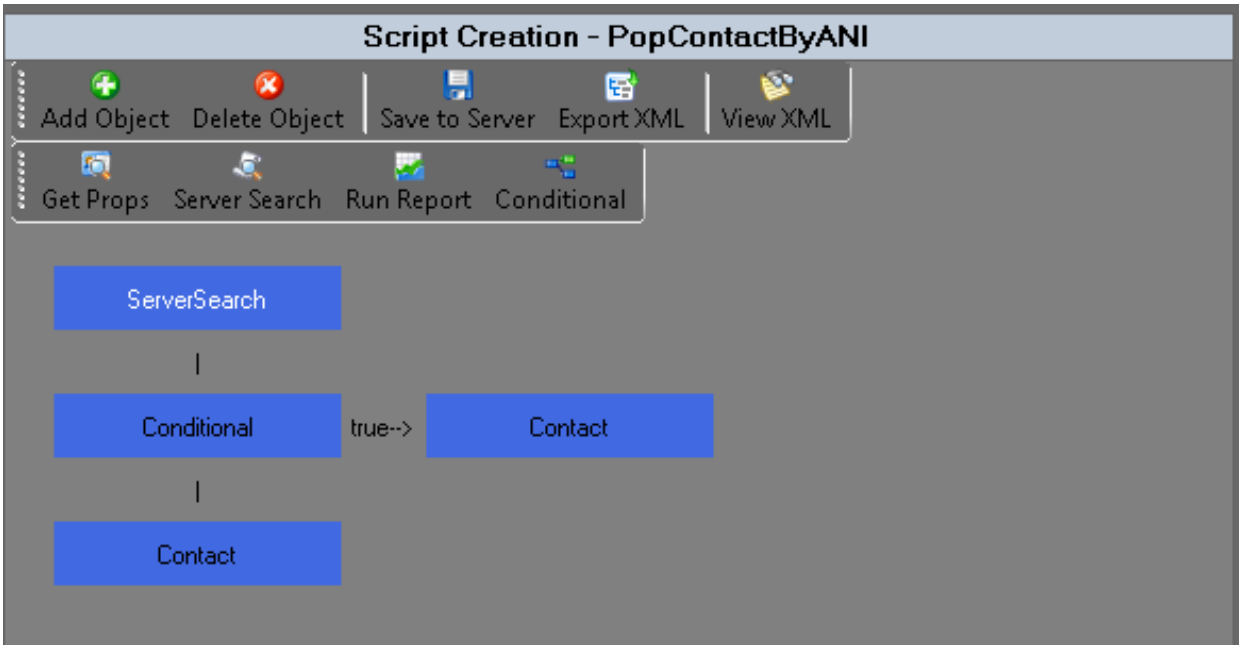
- PopContactByANI
- PopIncidentByRefNo
- PopNewIncidentWithContactByANI

### PopContactByANI

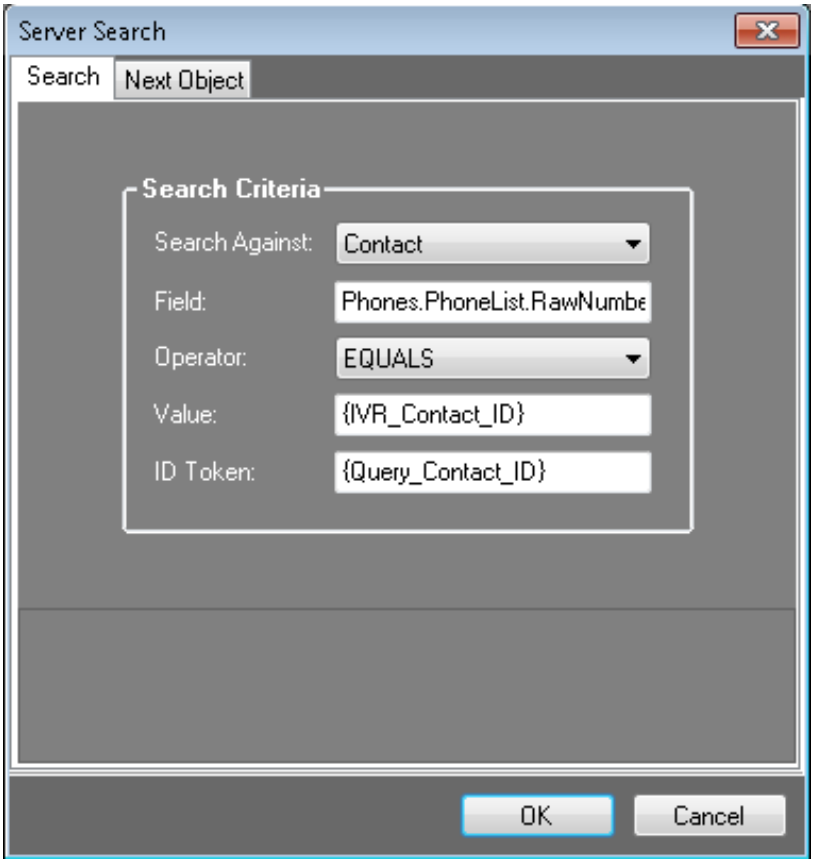
The PopContactByANI template creates a script that searches the server for a contact record and then opens the record for editing.



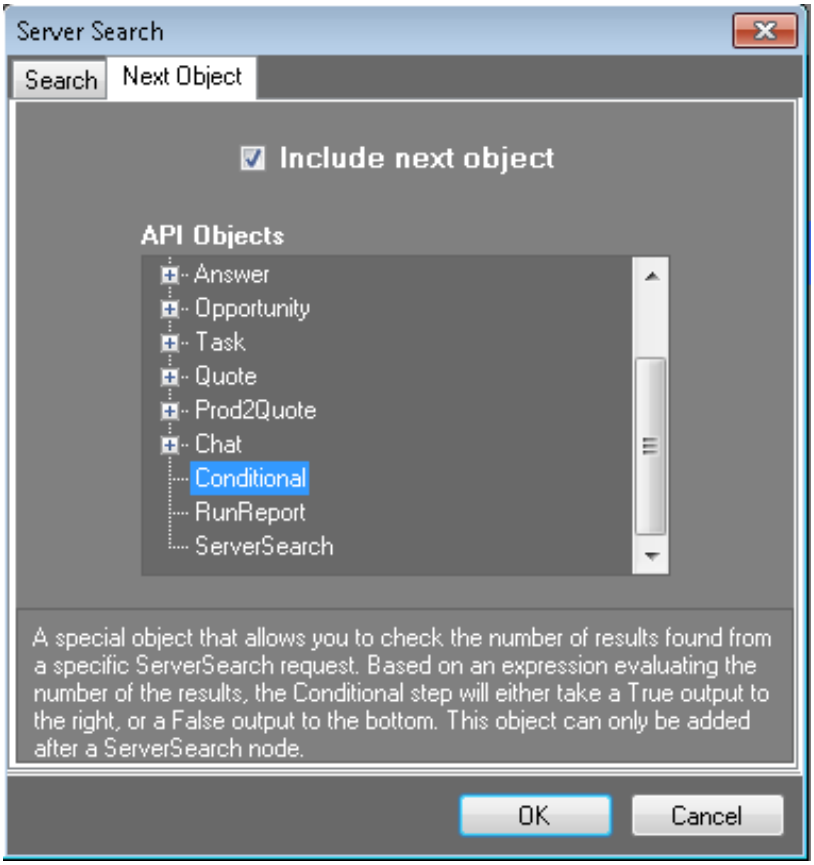

When you click the Template Script **PopContactByANI**, this window opens.



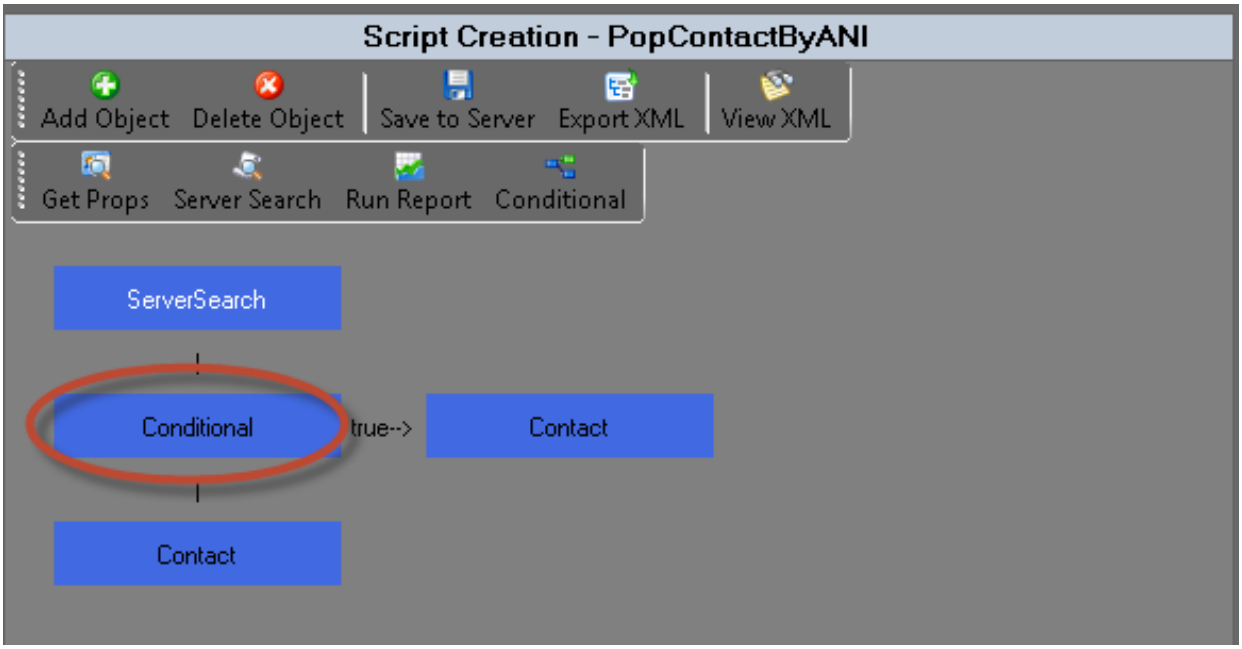

The first box is the **ServerSearch**. This **ServerSeach** searches against the **Contact** list for the **Phone Number** of the current interaction. The value of the `{IVR_Contact_ID}` comes from the IVR and the ID (or IDs) of the contact (or contacts) will be set to the `Query_Contact_ID` attribute.

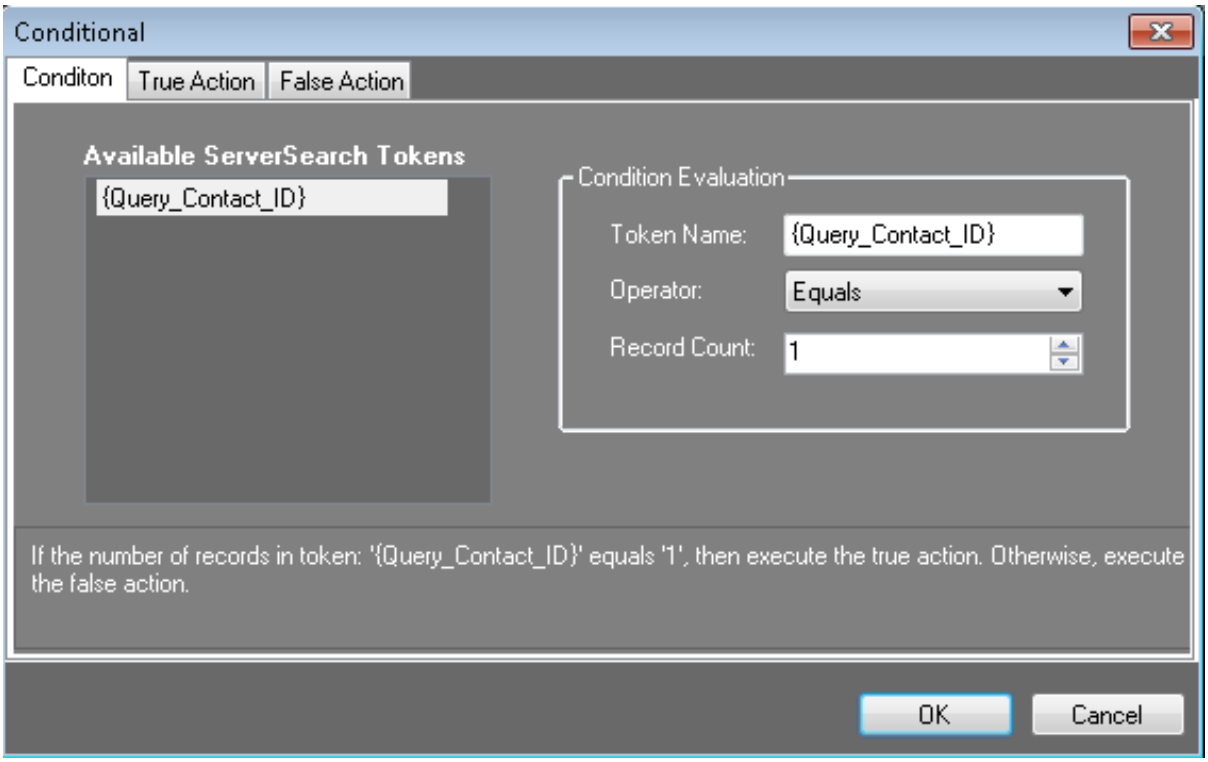



The **Next Object** tab is for telling the script what to do next. In this case, it is going to the **Conditional** object.

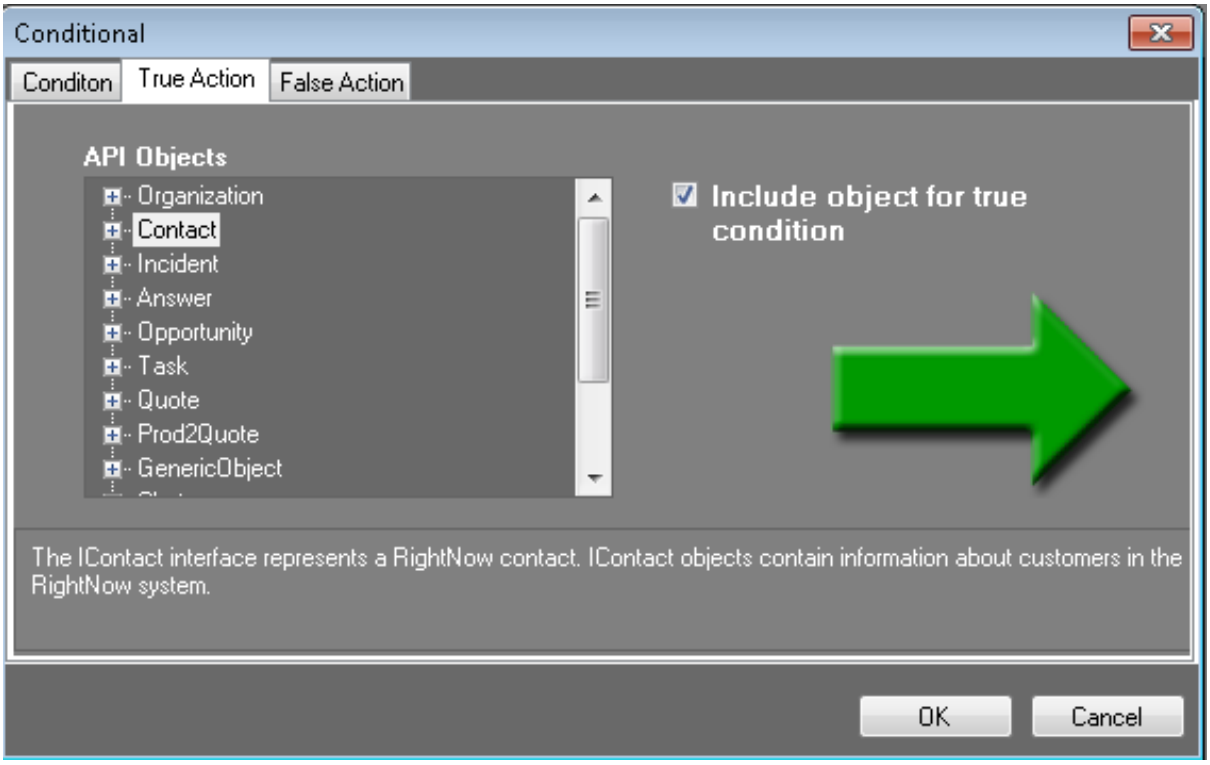



Next is the **Conditional** object.

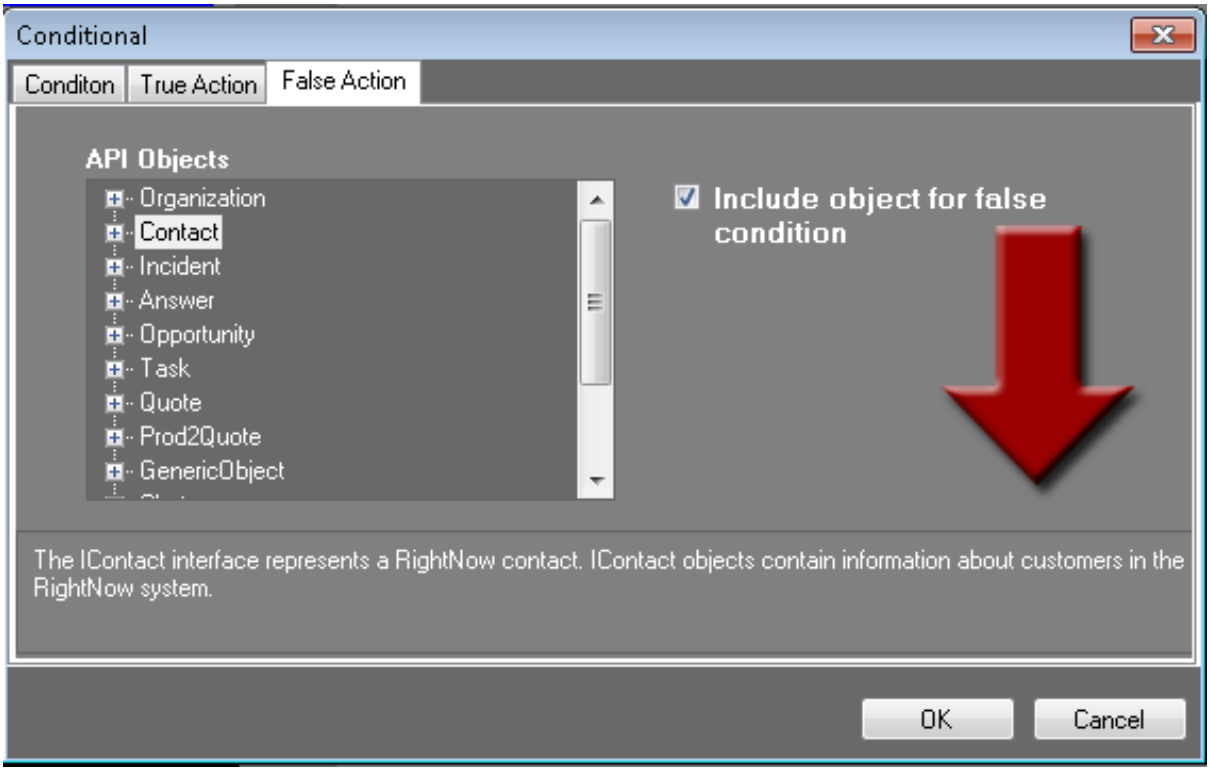


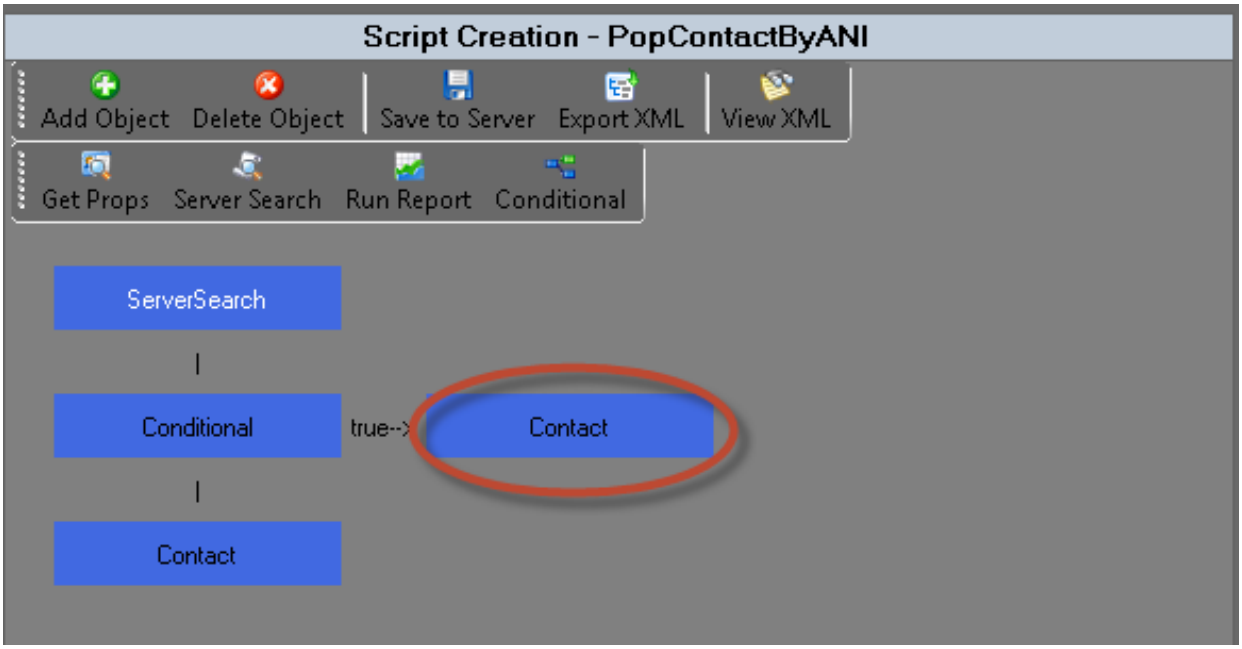

The **Condition** above states that: If the count of **Contact Id** = 1. This means that one record was found in the search.

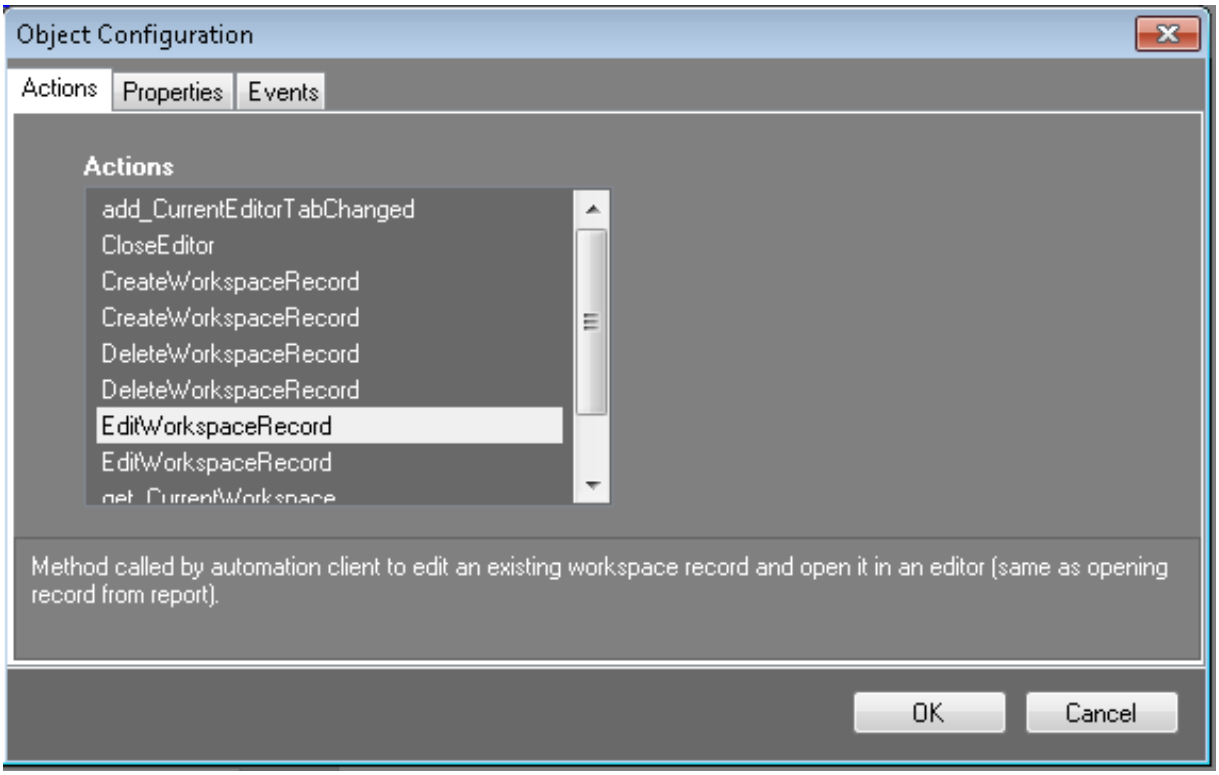



The **True Action** is: Then pop the **Contact** information.

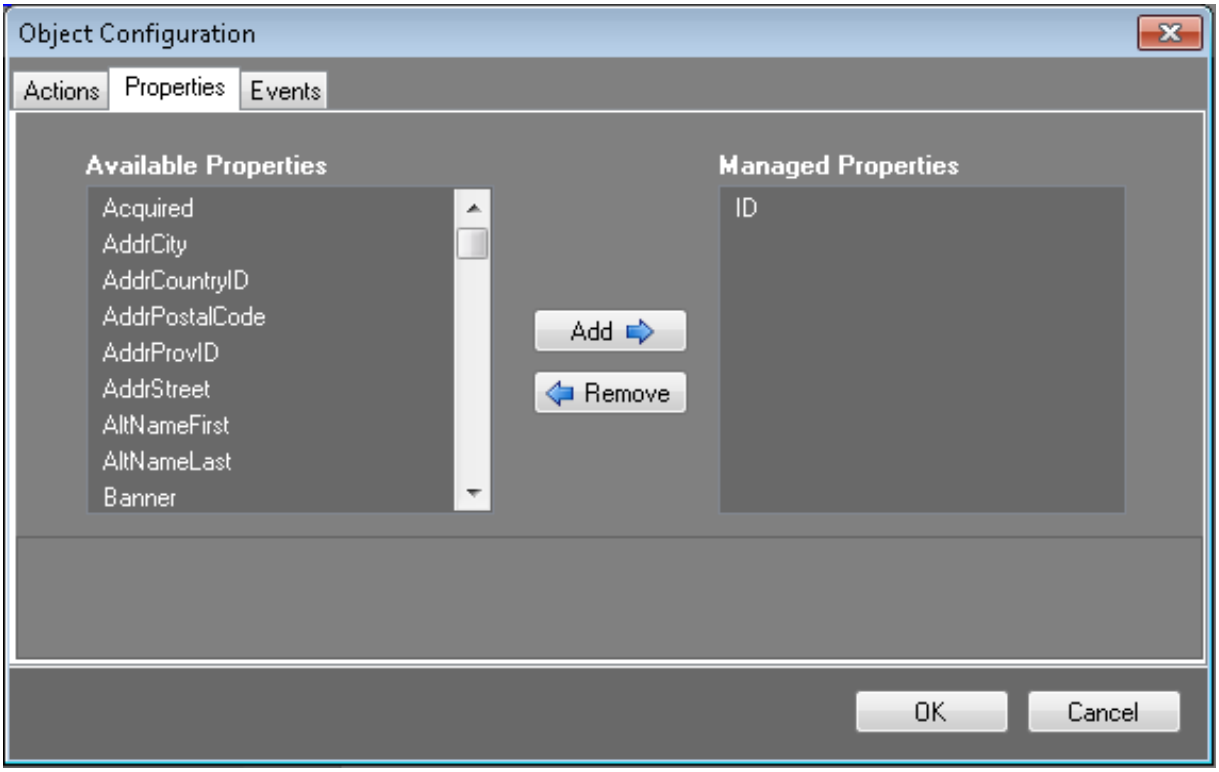



The **False Action** is: Else pop the **Contact** information. At first glance this seems to be doing the same thing whether True or False, but as you see in the next screen shot there is a **Contact** object if True and a different **Contact** object if False. These finish off the Then and Else of the condition.

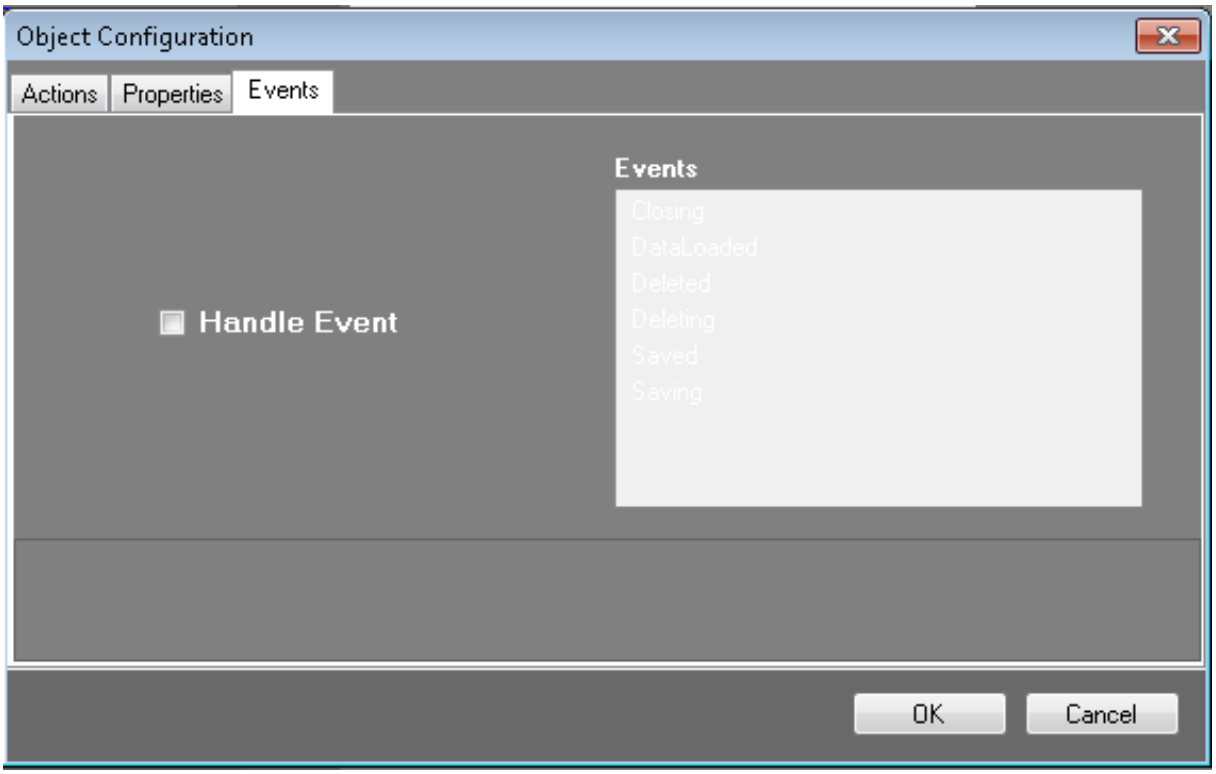



On the **Contact** object of the True side it is saying pop the record as an **Edit** record. It has found the **Contract** record for the person on the phone and it is popping it as an edit form.

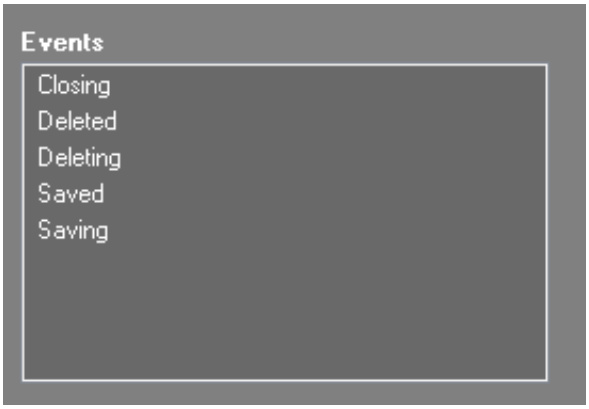


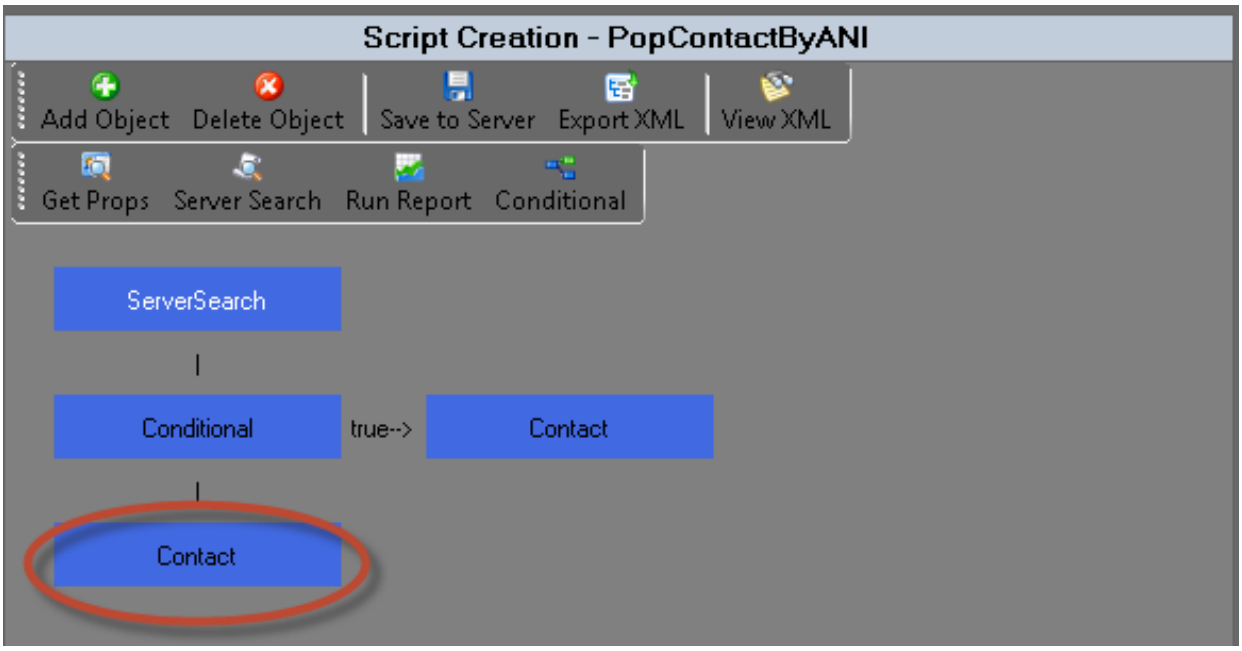

It is using the ID as the matching field.

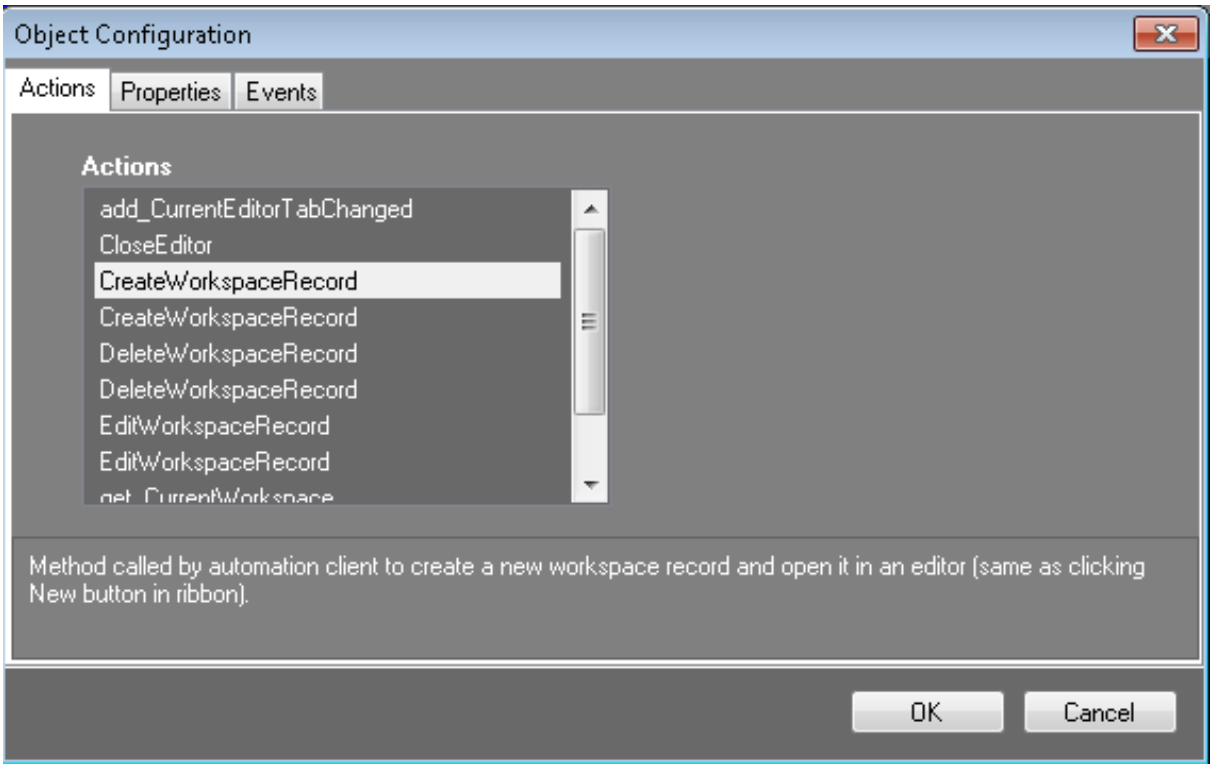



There are no Events chosen.

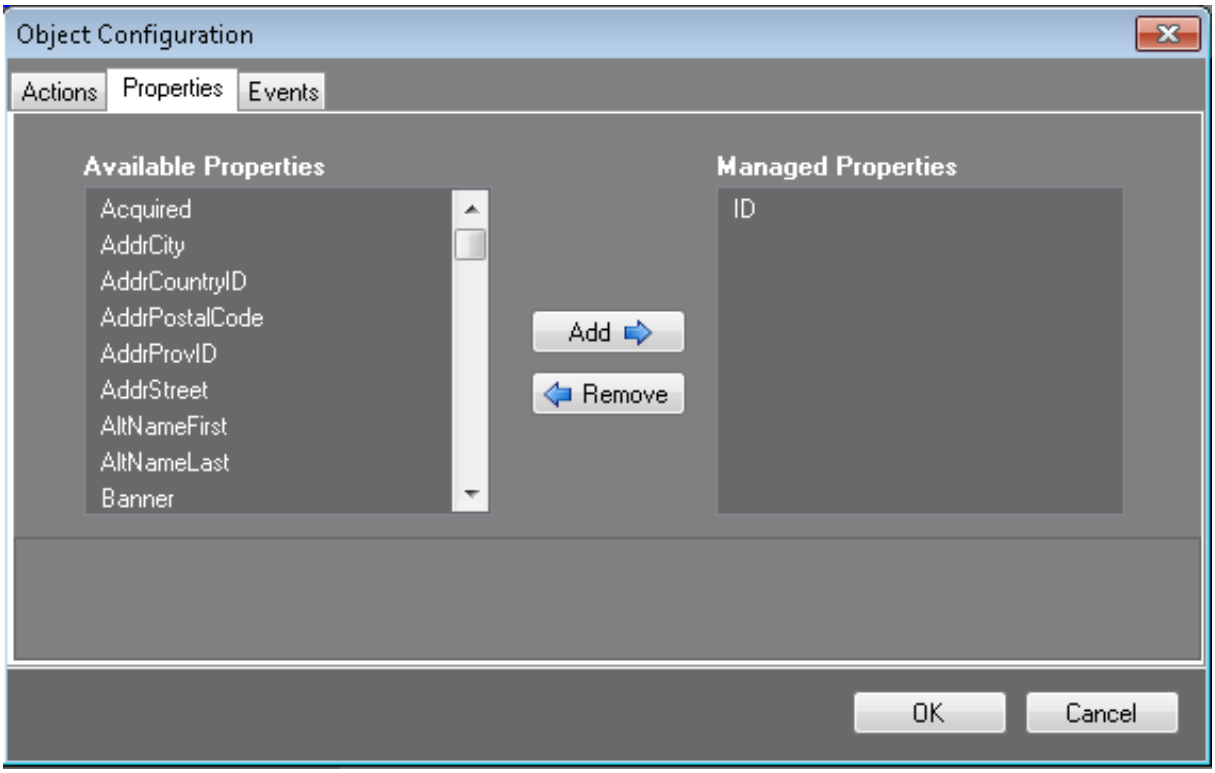
To have an **Event** happen after the action, click the **Handle Event** checkbox and the options become available. You can then choose what happens when the evet is fired from Oracle Service Cloud. These events come from the workspace record in Oracle Service Cloud.


Events
Closing
Deleted
Deleting
Saved
Saving

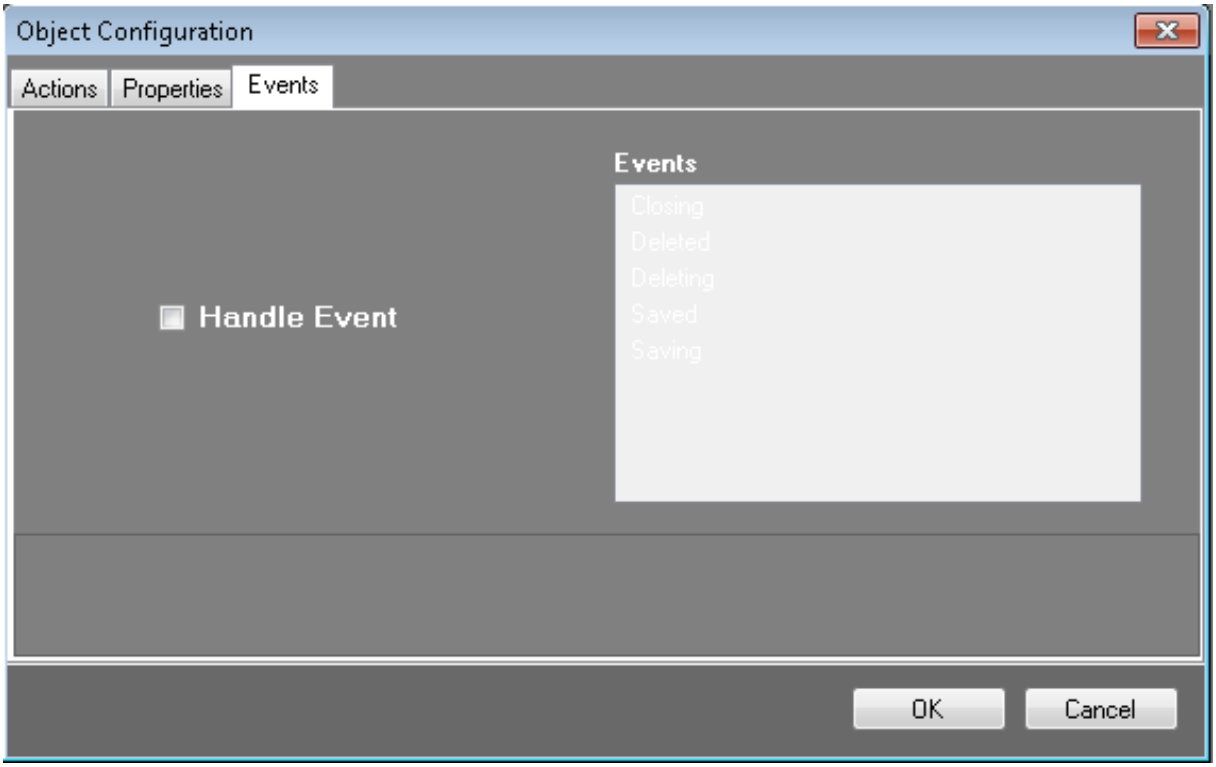

Script Creation - PopContactByANI
Add Object
Delete Object
Save to Server
Export XML
View XML
Get Props
Server Search
Run Report
Conditional
ServerSearch
Conditional
true-->
Contact
Contact


Object Configuration
Actions
Properties
Events
Actions
add_CurrentEditorTabChanged
CloseEditor
CreateWorkspaceRecord
CreateWorkspaceRecord
DeleteWorkspaceRecord
DeleteWorkspaceRecord
EditWorkspaceRecord
EditWorkspaceRecord
Method called by automation client to create a new workspace record and open it in an editor (same as clicking New button in ribbon).
OK
Cancel

On the **False** side it is saying to pop a **New** record for the **Contact** form. This is because it did not find the contact in the current **Contact** data and needs to create a new **Contact**.




It uses the ID from the call as the ID for this new record.



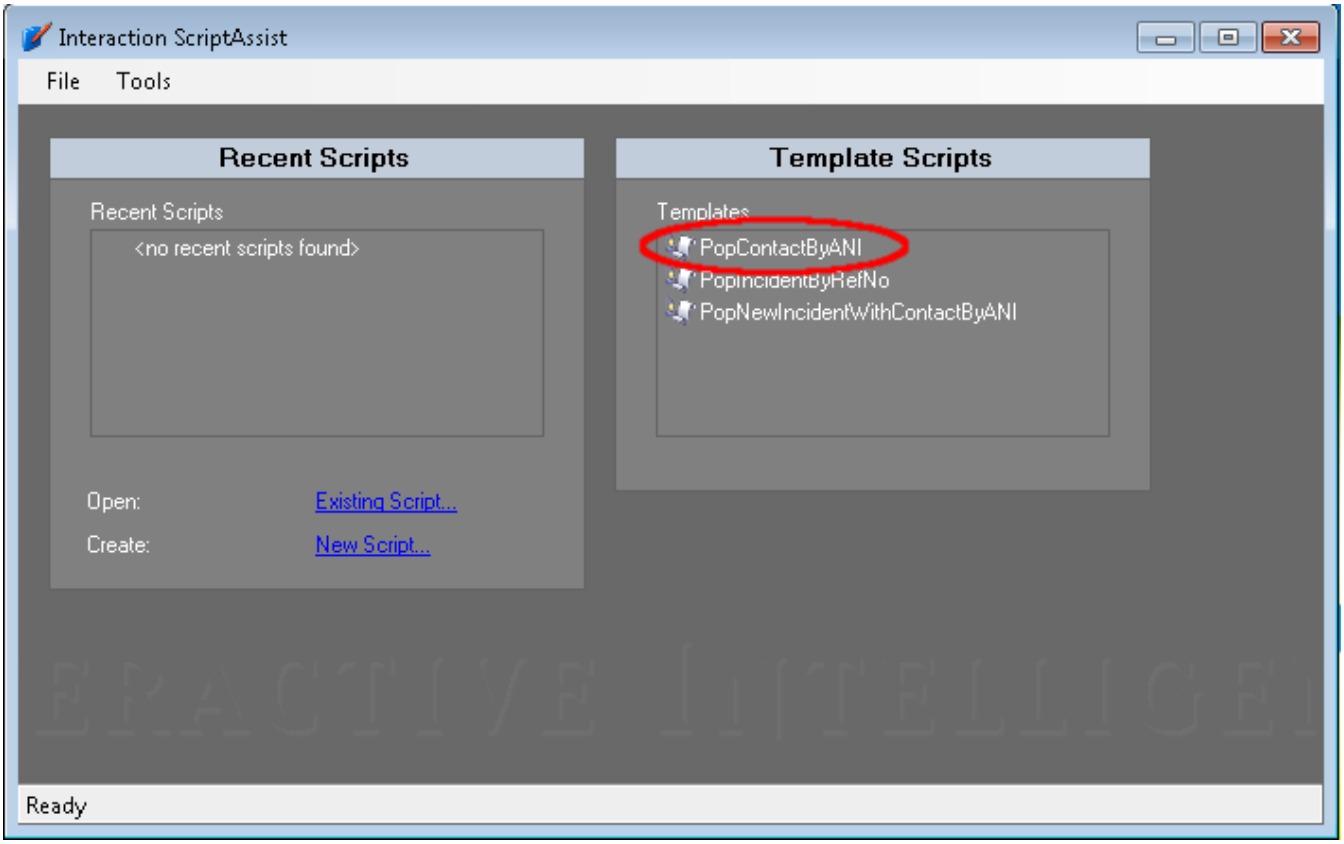

There are no Events chosen.

Summary of the script:

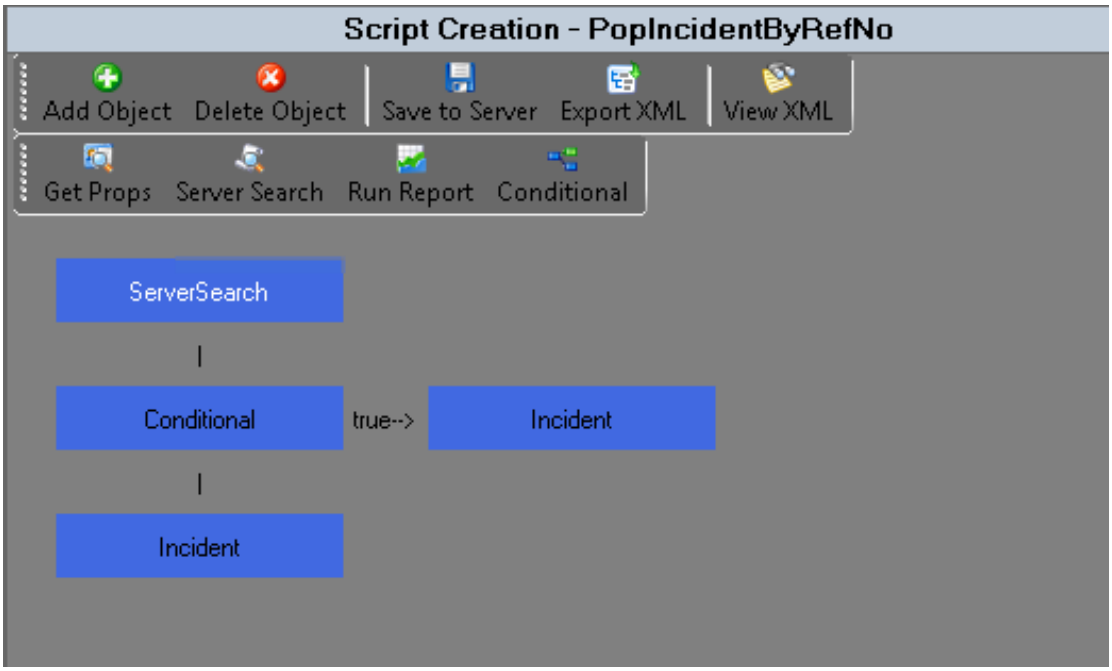
1. Gets the ANI from the IVR.
2. Looks in the **Contact** data for a match to the caller.
3. If it finds a match it sends that information to the Oracles Service Cloud client and pops the contact on the screen.
4. If it doesn't find a match it teslls Oracle Service Cloud client to pop a new **Contact** record.

## PopIncidentByRefNo

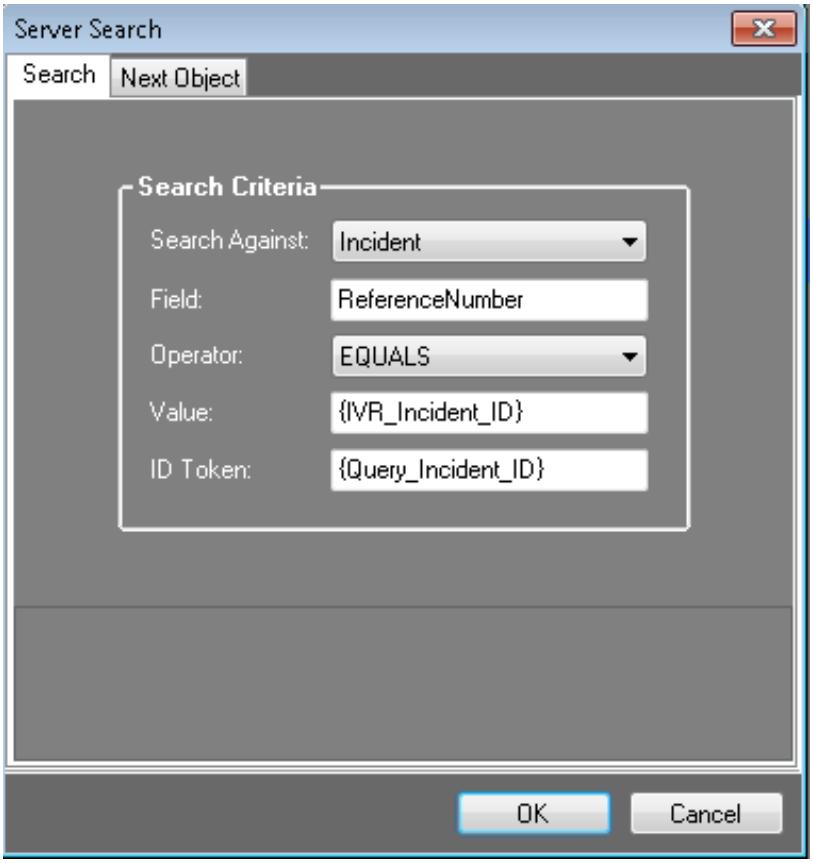
The PopIncidentByRefNo template creates a script that searches the server for a specific incident record and then opens the record for editing.



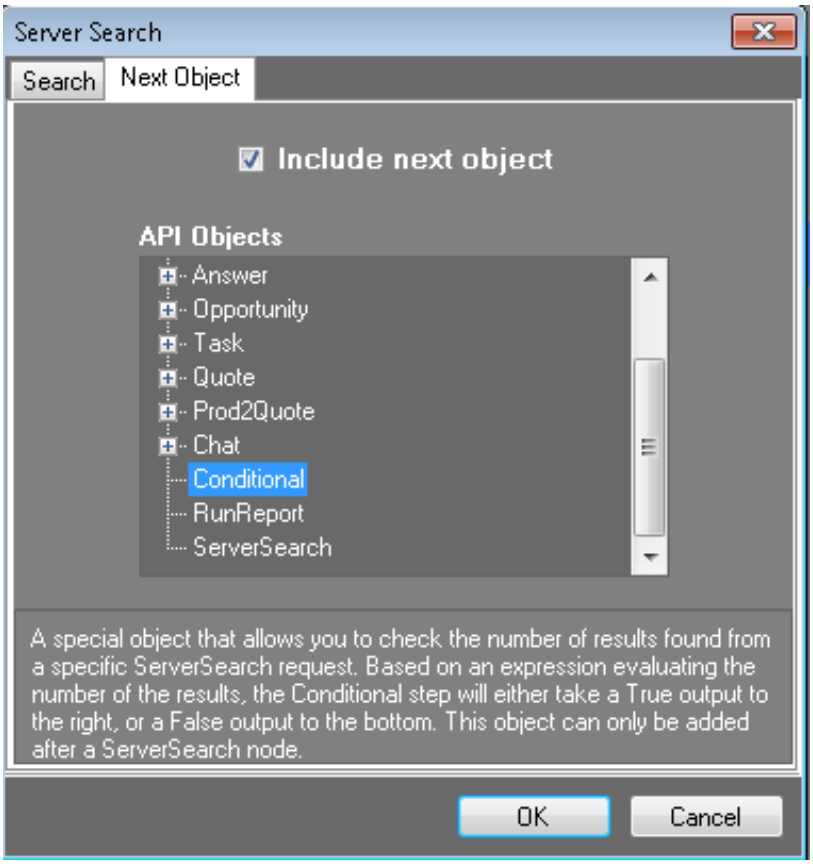

When you click the Template Script **PopIncidentByRefNo**, this window opens.



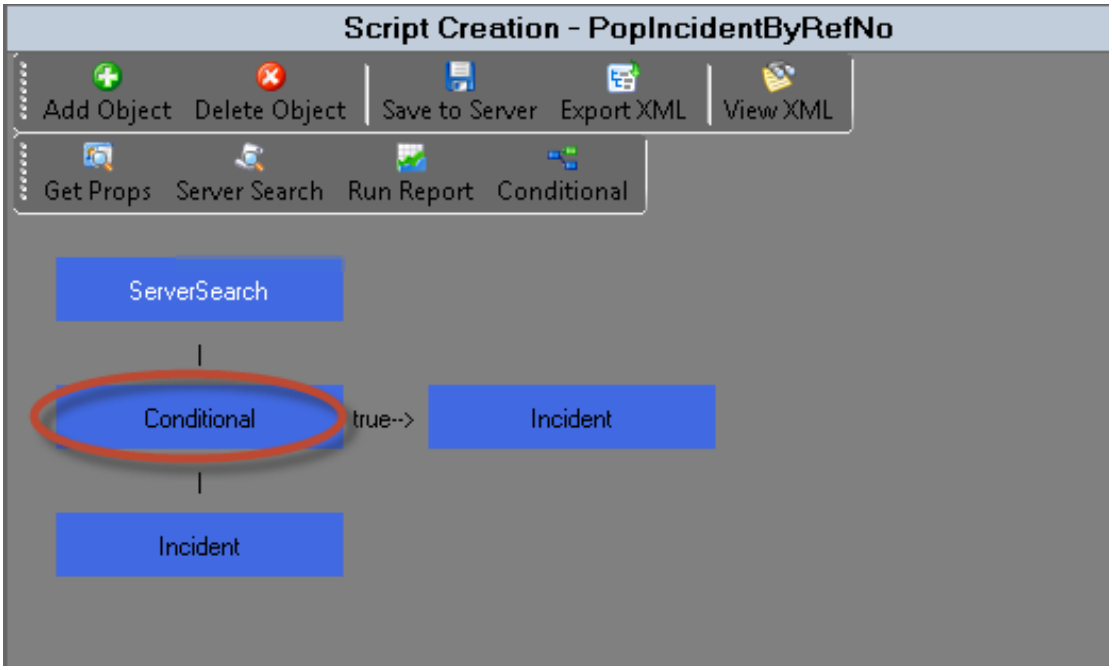

The first box is the **ServerSearch**. This **ServerSearch** searches against the **Incident** objects for the **Reference Number** of the current interaction. The value of the `{IVR_Incident_ID}` comes from the IVR and ID of the interaction will be set to the `Query_Incident_ID` attribute.

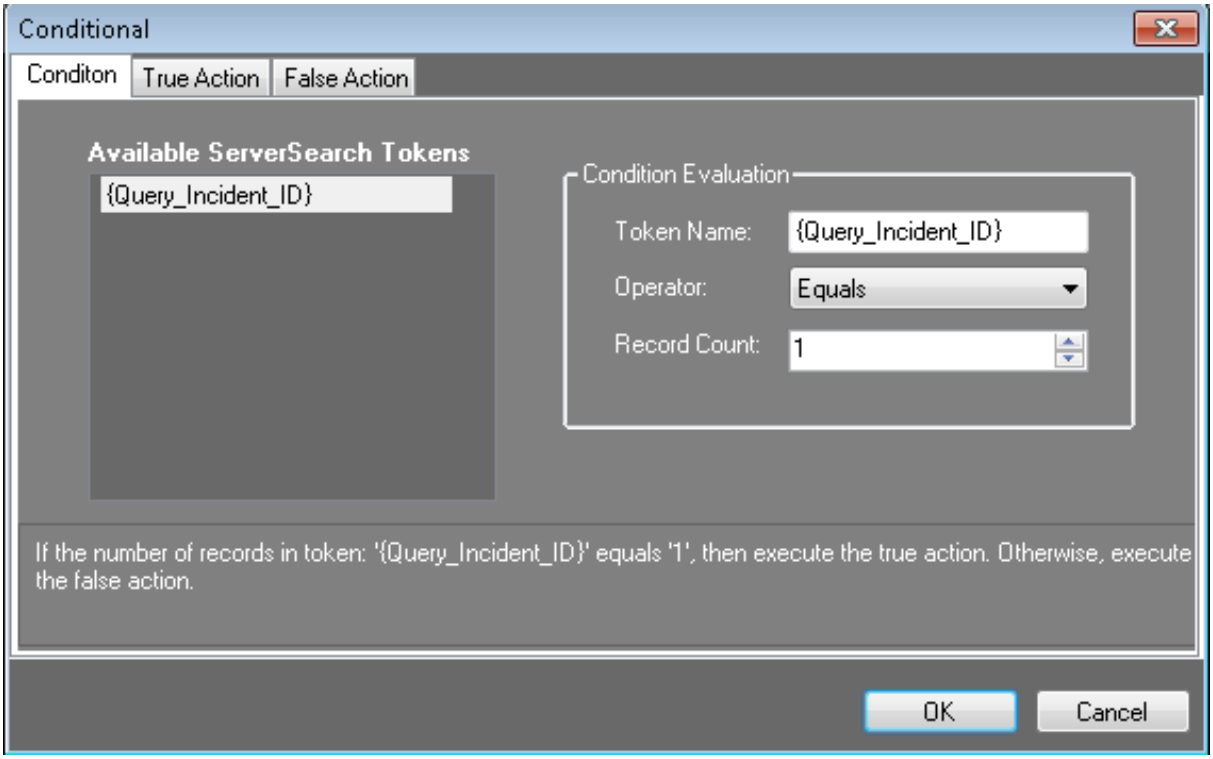



The **Next Object** tab is for telling the script what to do next. In this case, it is going to the **Conditonal** object.

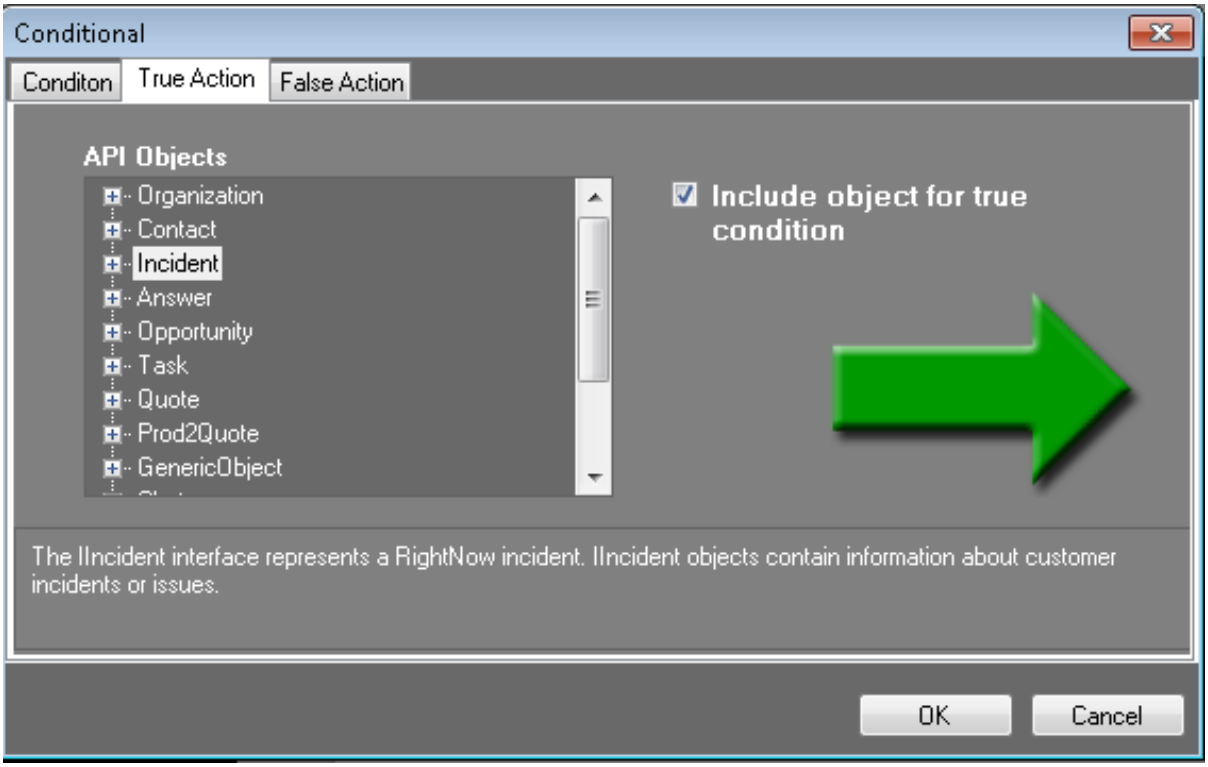



Next is the **Conditional** object.

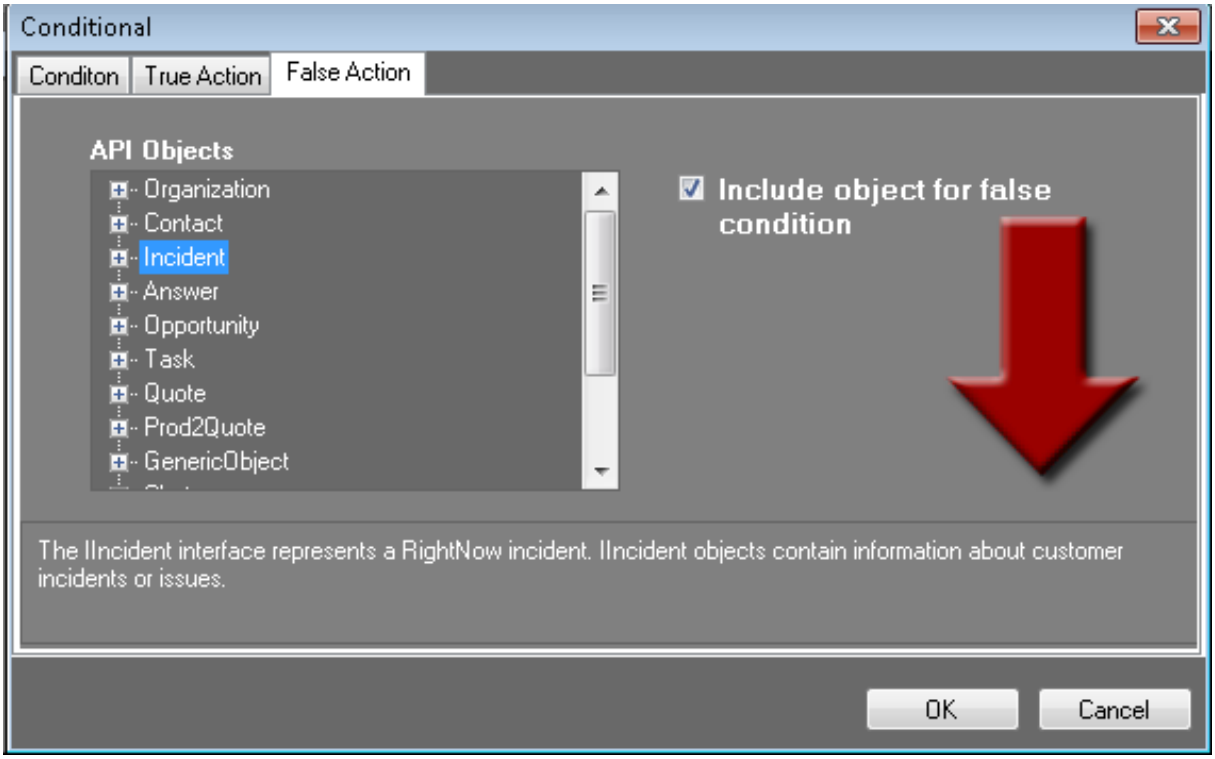


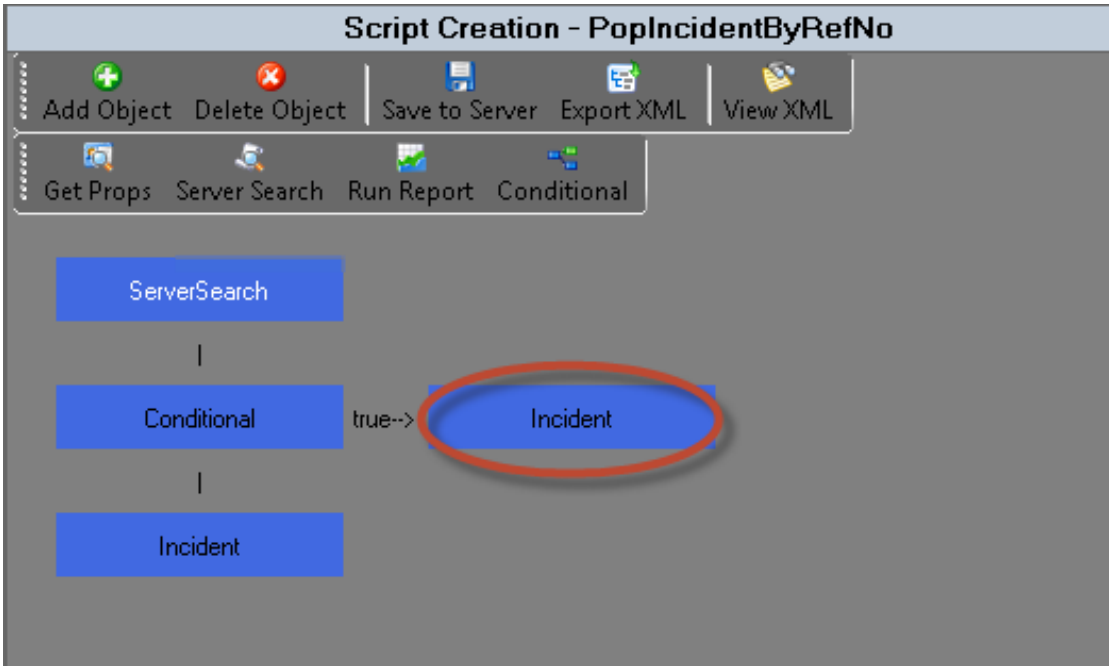

The **Condition** above states that: **If the count of Incident ID = 1**. This means that one record was found.

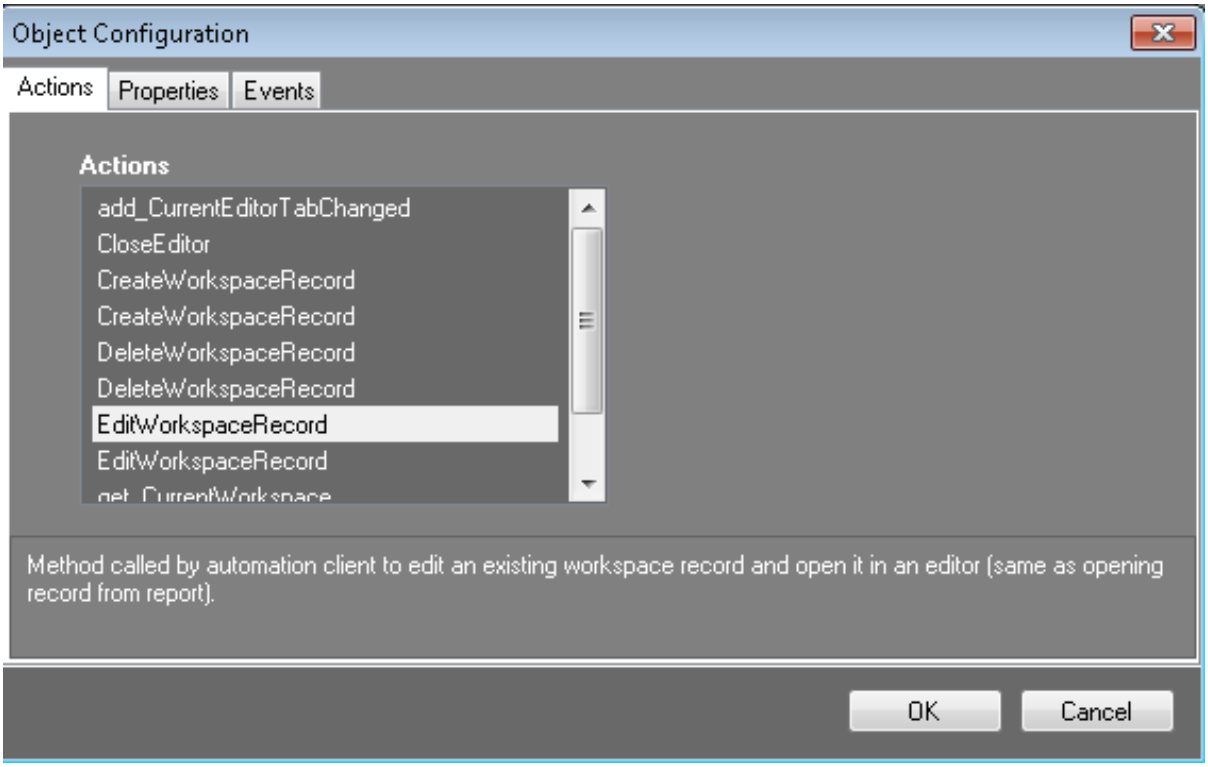



The **True Action** is: Then pop the **Incident** information.

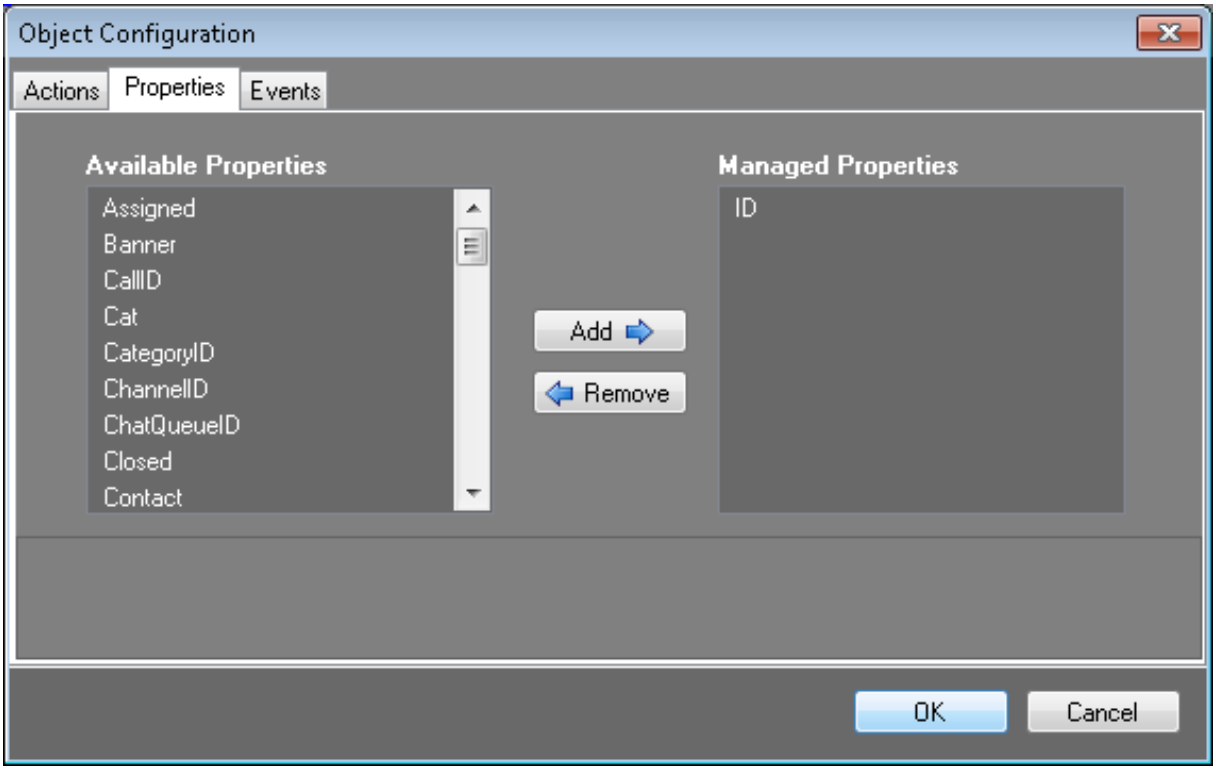



The **False Action** is: Else pop the **Incident** information. At first glance, this seems to be doing the same thing whether True or False, but as you see in the next screen shot there is an **Incident** object if True and a different **Incident** object if False. These finish off the Then and Else of the condition.

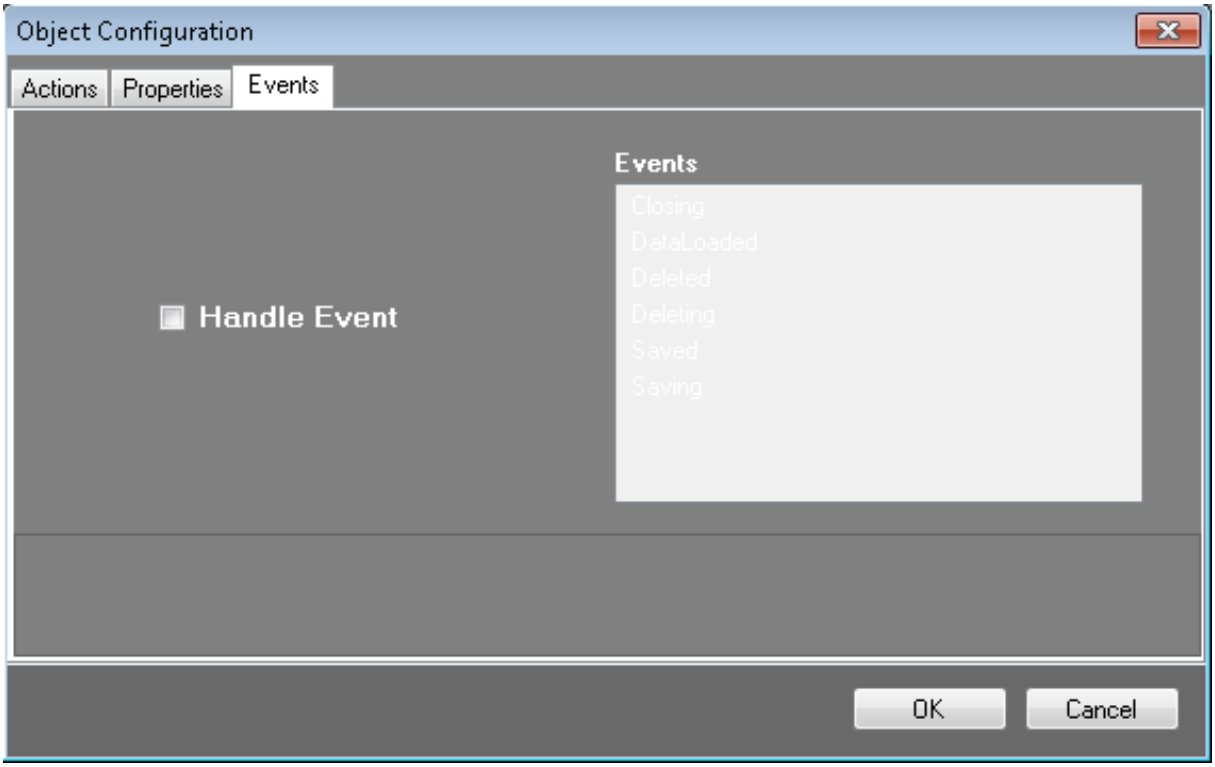



On the **Incident** object of the True side, it is saying pop the record as an **Edit** record. So it has found the **Incident** record for the person on the phone and it is popping it as an edit form.

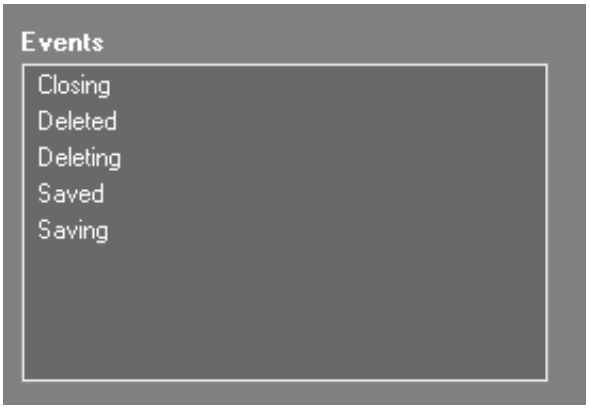


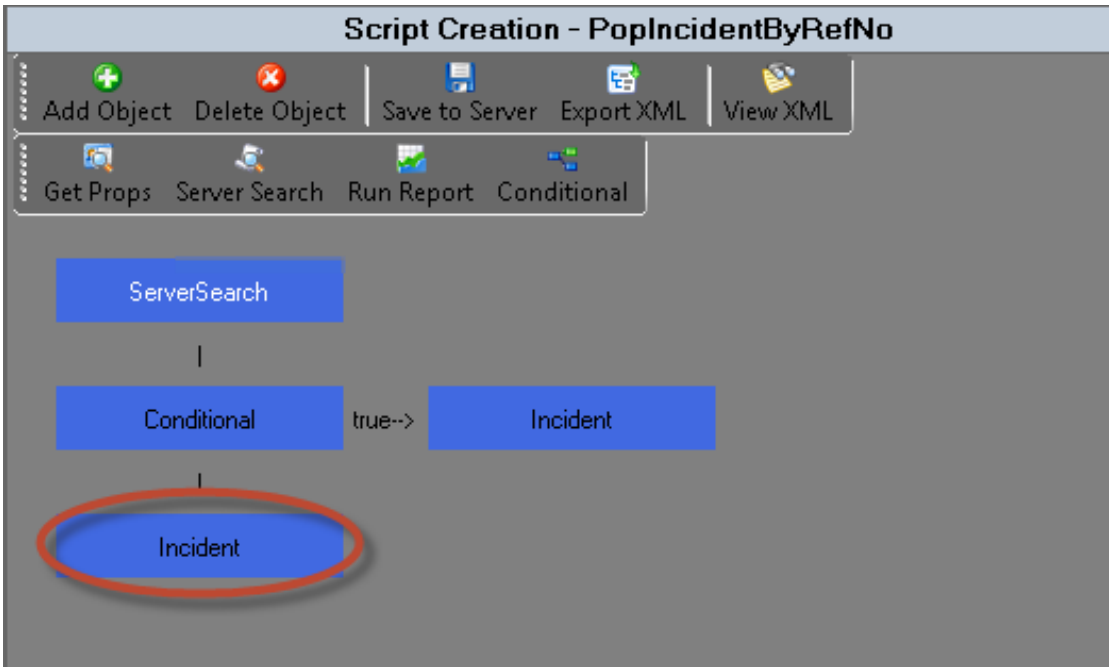

It uses the ID as the matching field.

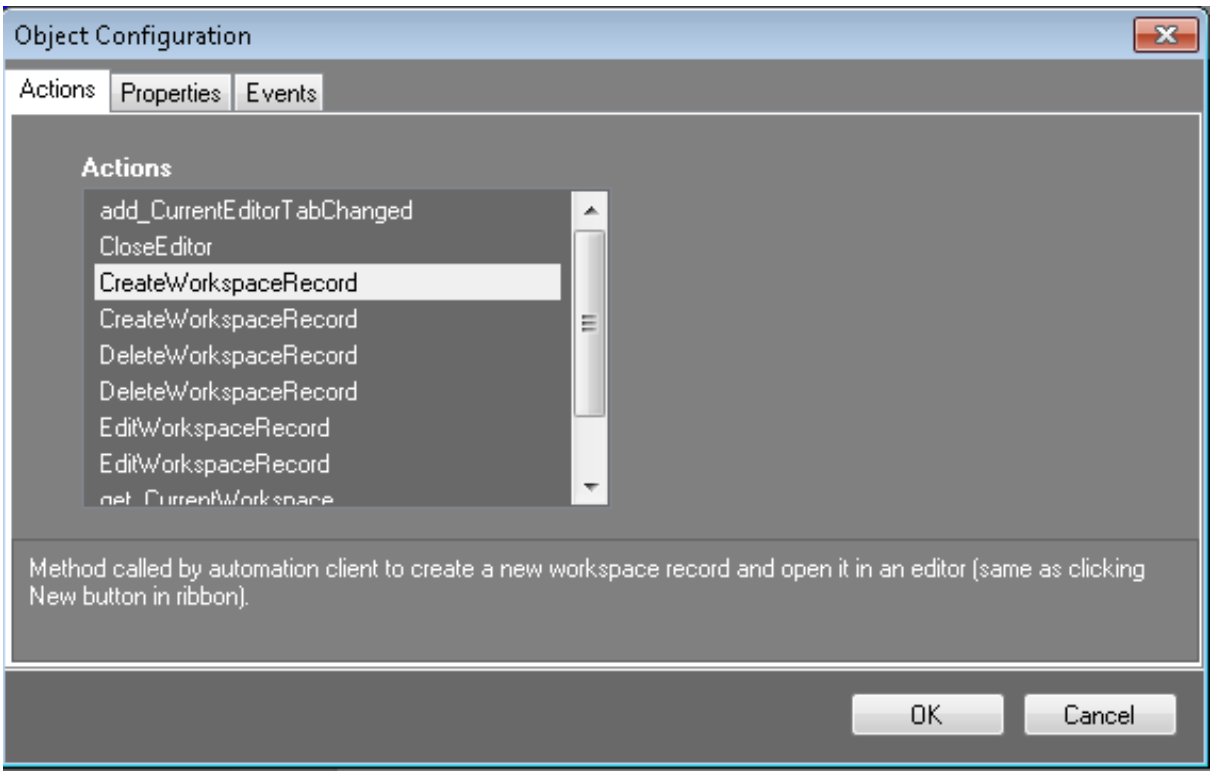



There are no **Events** chosen.

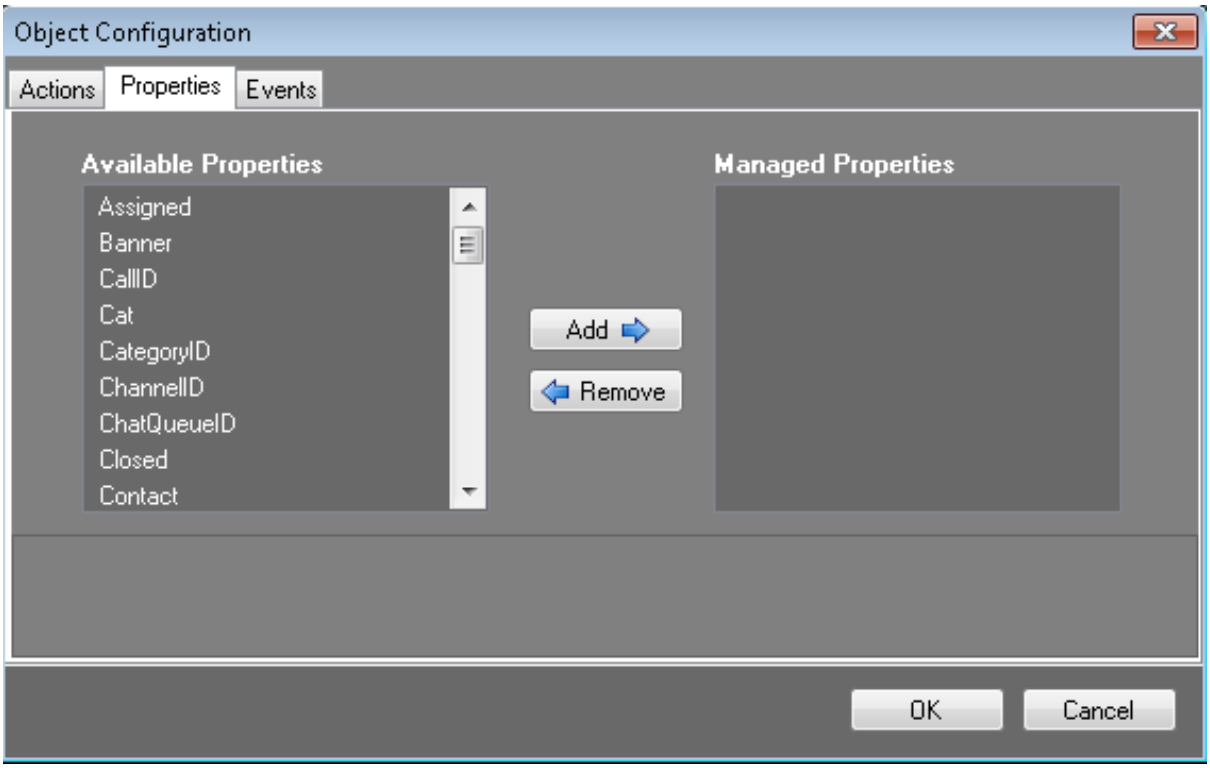
To have an **Event** happen after the action, click the **Handle Event** checkbox and the options become available. You can then choose what happens when the evet is fired from Oracle Service Cloud. These events come from the workspace record in Oracle Service Cloud.


Events
Closing
Deleted
Deleting
Saved
Saving

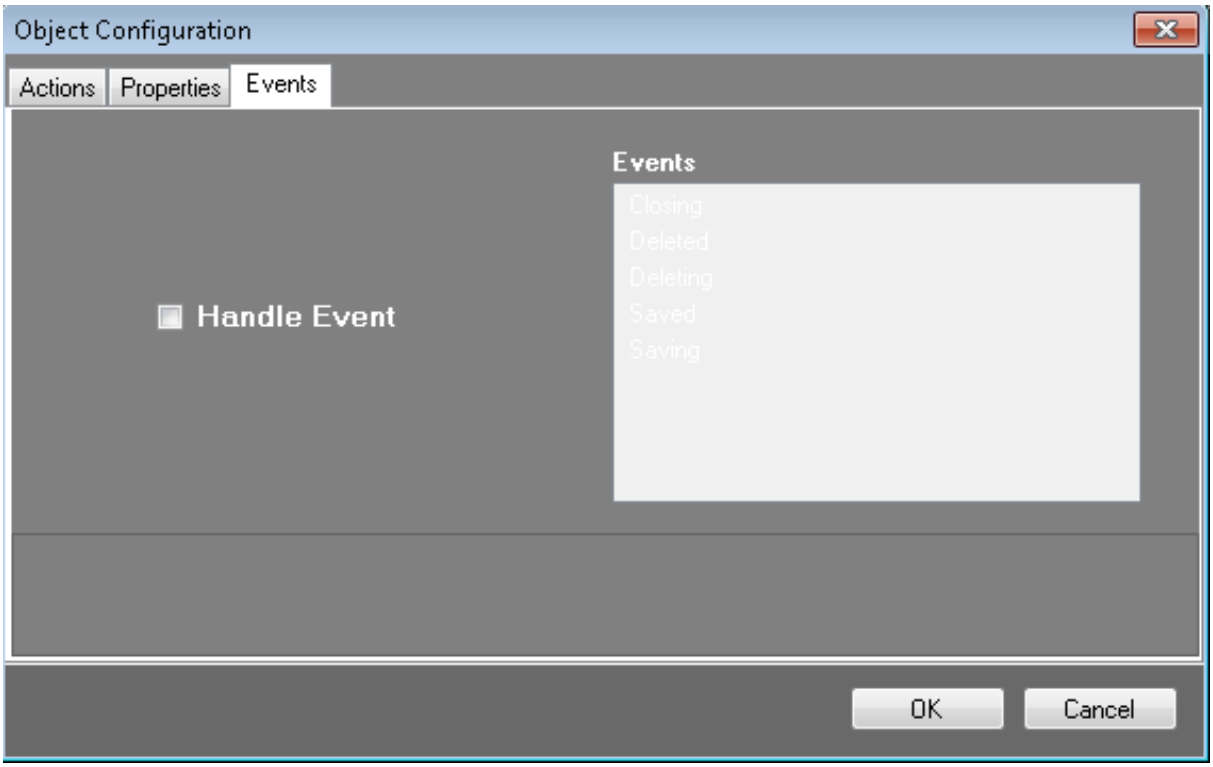

Script Creation - PopIncidentByRefNo
Add Object
Delete Object
Save to Server
Export XML
View XML
Get Props
Server Search
Run Report
Conditional
ServerSearch
Conditional
true-->
Incident
Incident


Object Configuration
Actions
Properties
Events
Actions
add_CurrentEditorTabChanged
CloseEditor
CreateWorkspaceRecord
CreateWorkspaceRecord
DeleteWorkspaceRecord
DeleteWorkspaceRecord
EditWorkspaceRecord
EditWorkspaceRecord
Method called by automation client to create a new workspace record and open it in an editor (same as clicking New button in ribbon).
OK
Cancel

On the False side, it is saying to pop a **New** record for the **Incident** form. This is because it did not find the contact in the current **Incident** data and needs to create a new **Incident**.




Since it did not find a contact, there is nothing passed to the new record.



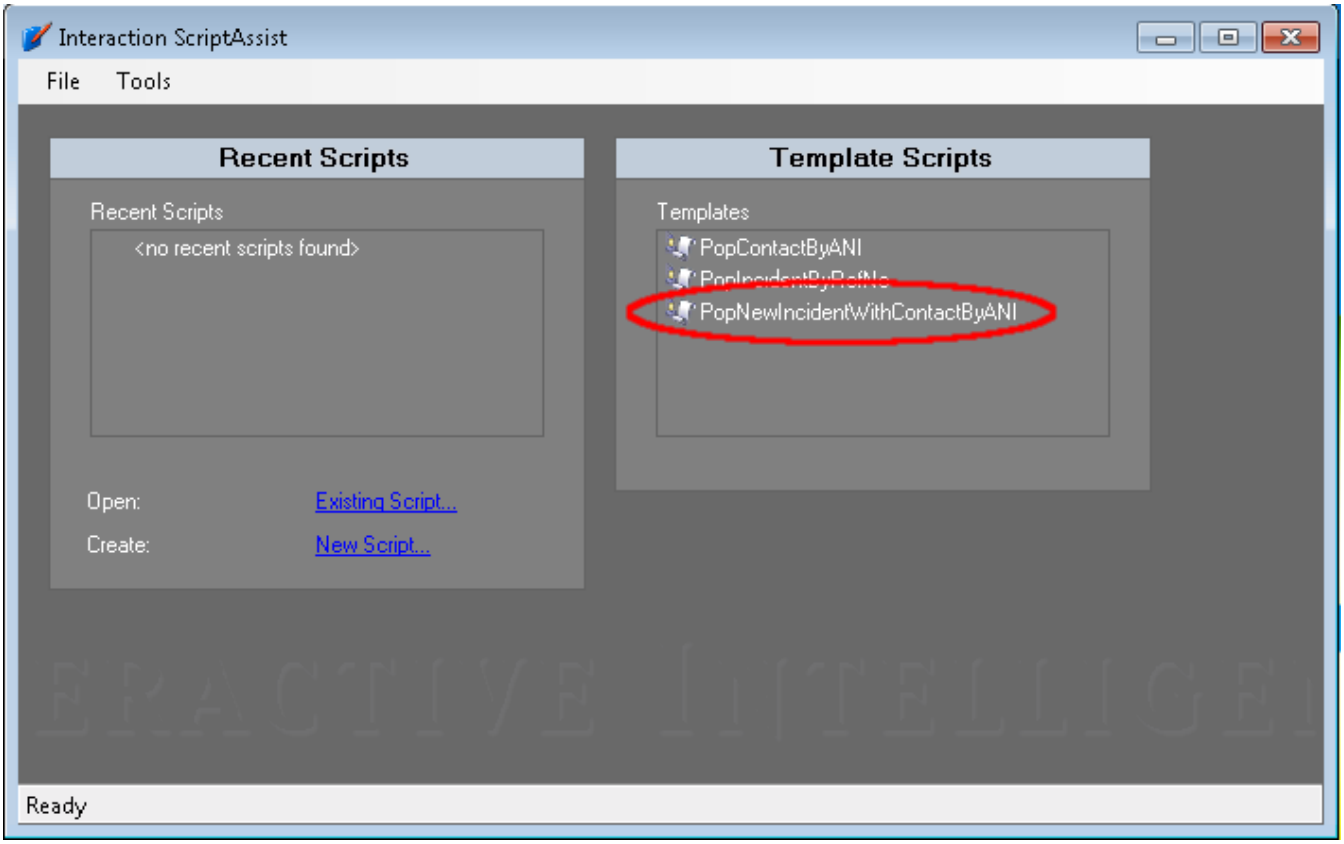

There are no **Events** chosen.

Summary of the script:

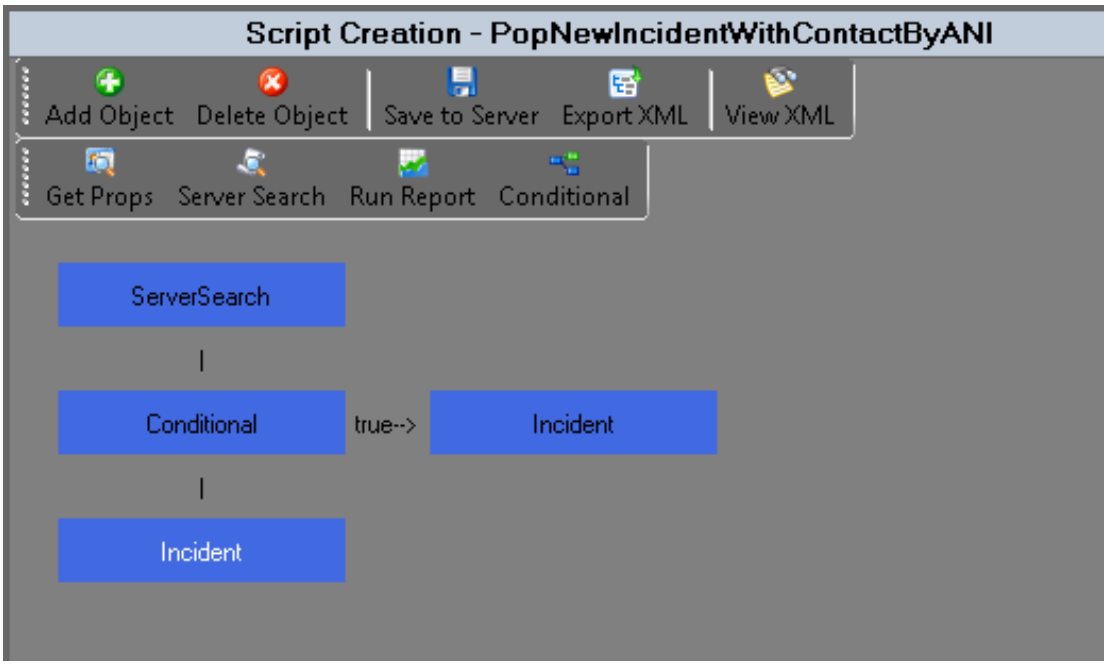
1. Gets the ANI from the IVR.
2. Looks in the **Incident** data for a match to the caller.
3. If it finds a match it sends that information to the Oracles Service Cloud client and pops the **Incident** on the screen.
4. If it doesn't find a match it teslls Oracle Service Cloud client to pop a new **Incident** record.

## PopNewIncidentWithContactByANI

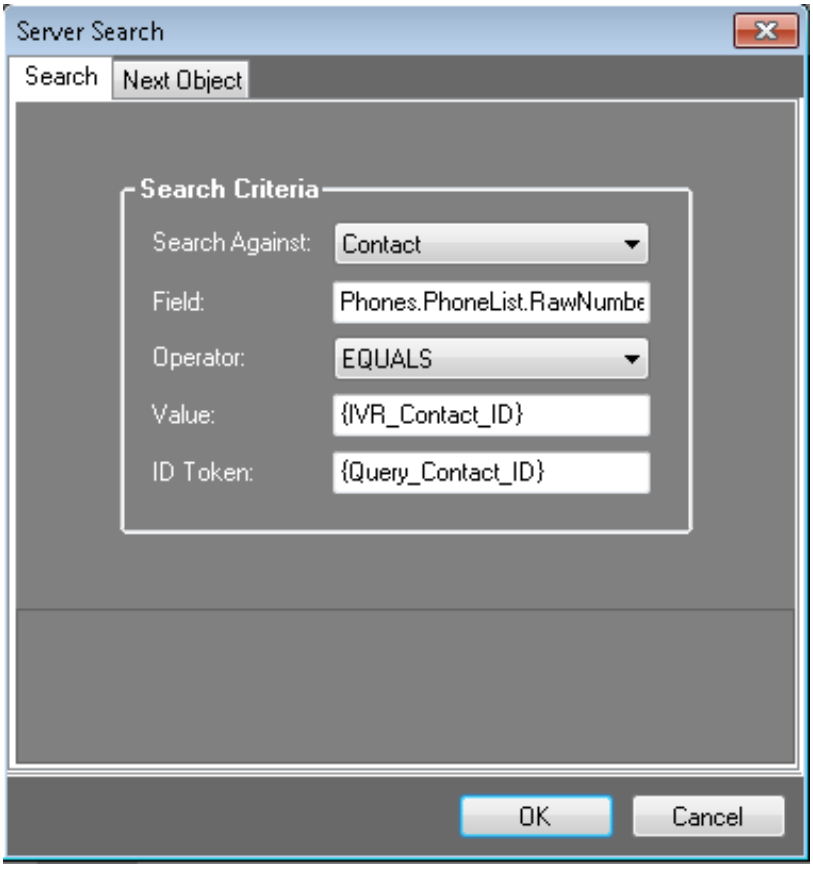
The PopNewIncidentWithContactByANI template creates a script that searches the server for a specific contact record and then creates an incident based on the record.



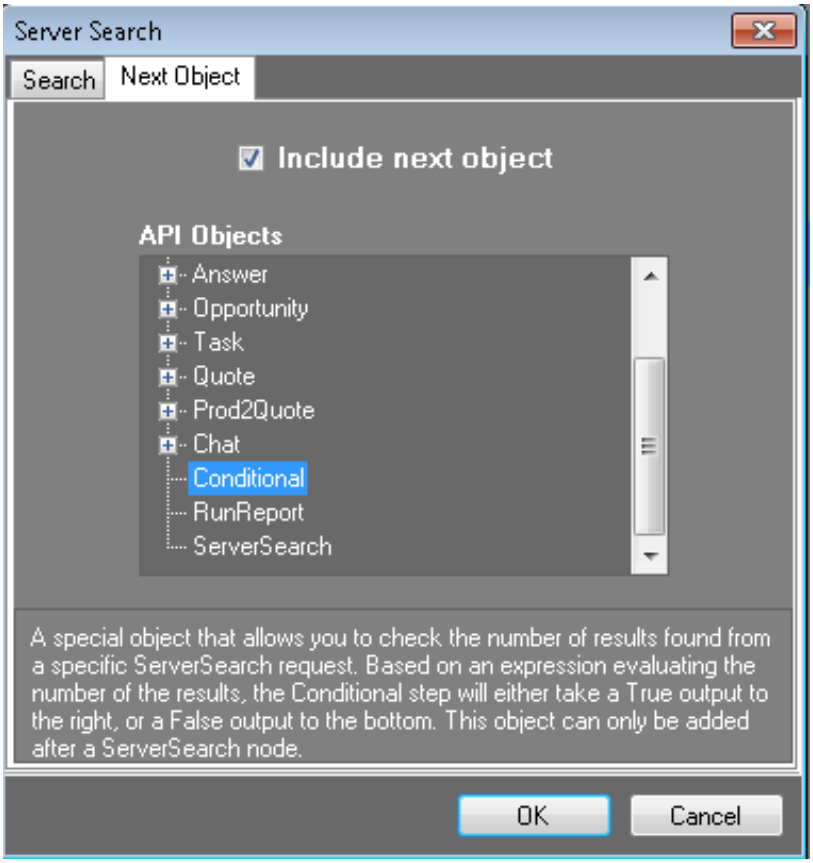

When you click the Template Script **PopNewIncidentWintContactByANI**, this window opens.



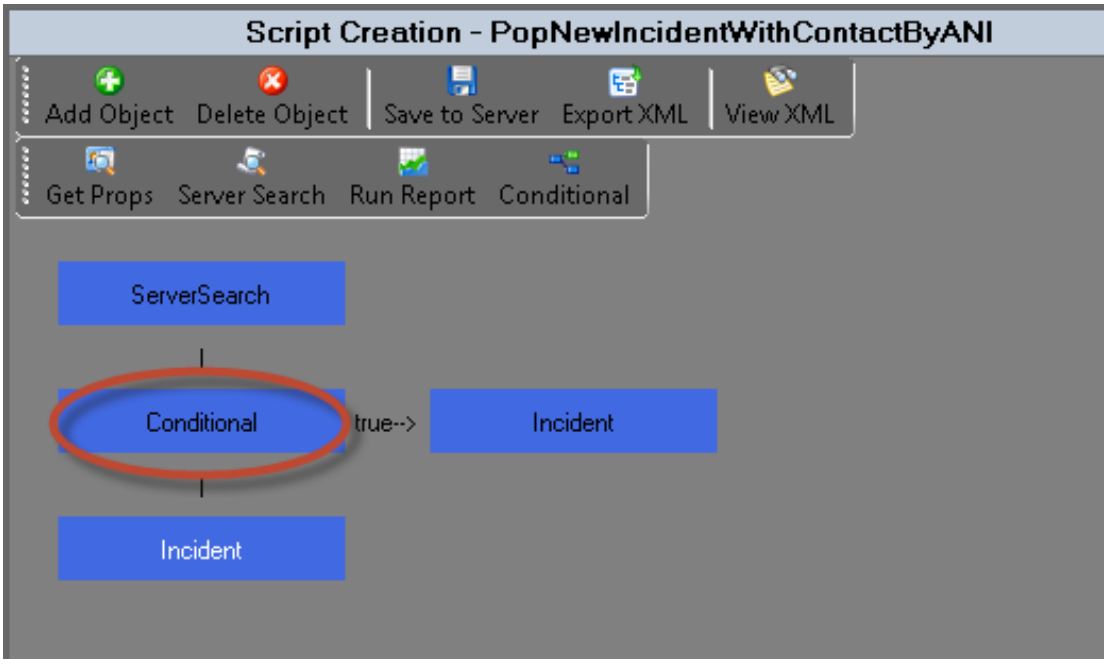

The first box is the **ServerSearch**. As you see below, this **ServerSearch** searches against the **Contact** data for the **Phone Number** of the current interaction. The value of the `{IVR_Contact_ID}` comes from the IVR and ID of the **Contact** will be set to the `Query_Contact_ID` attribute.

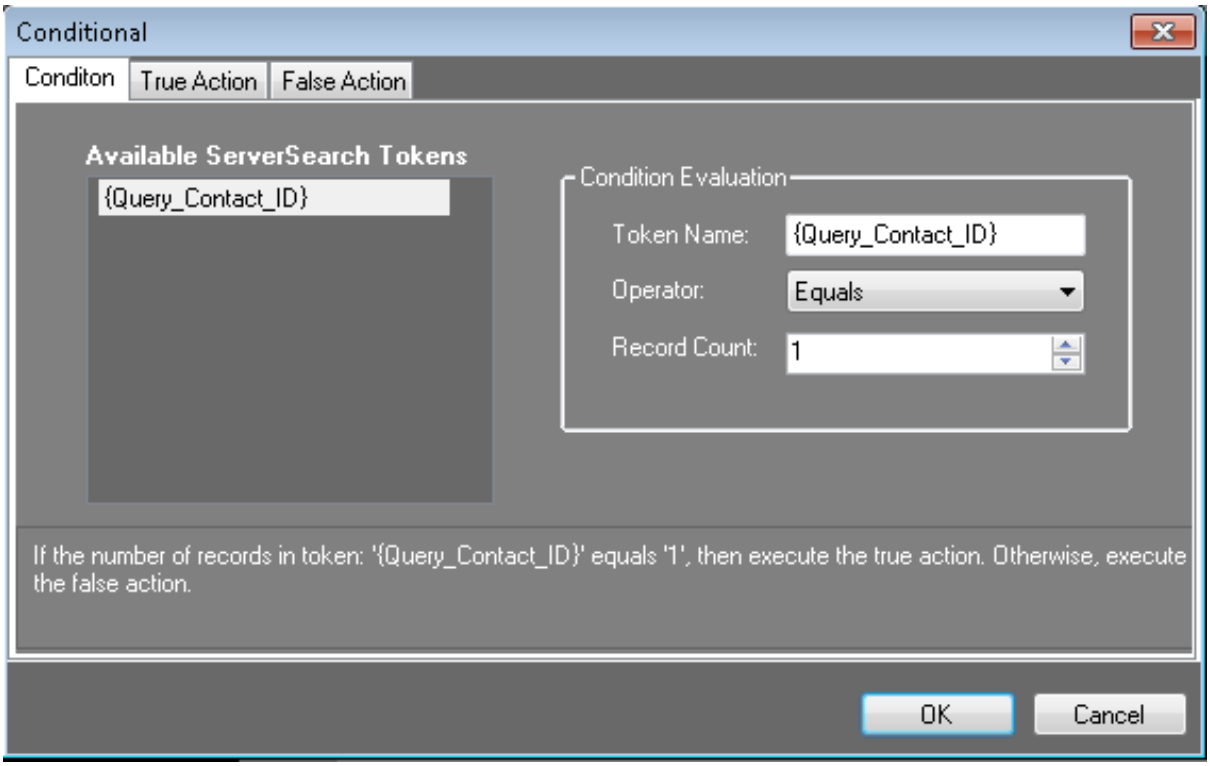



The **Next Object** tab is for telling the script what to do next. In this case, it is going to the **Conditional** object.

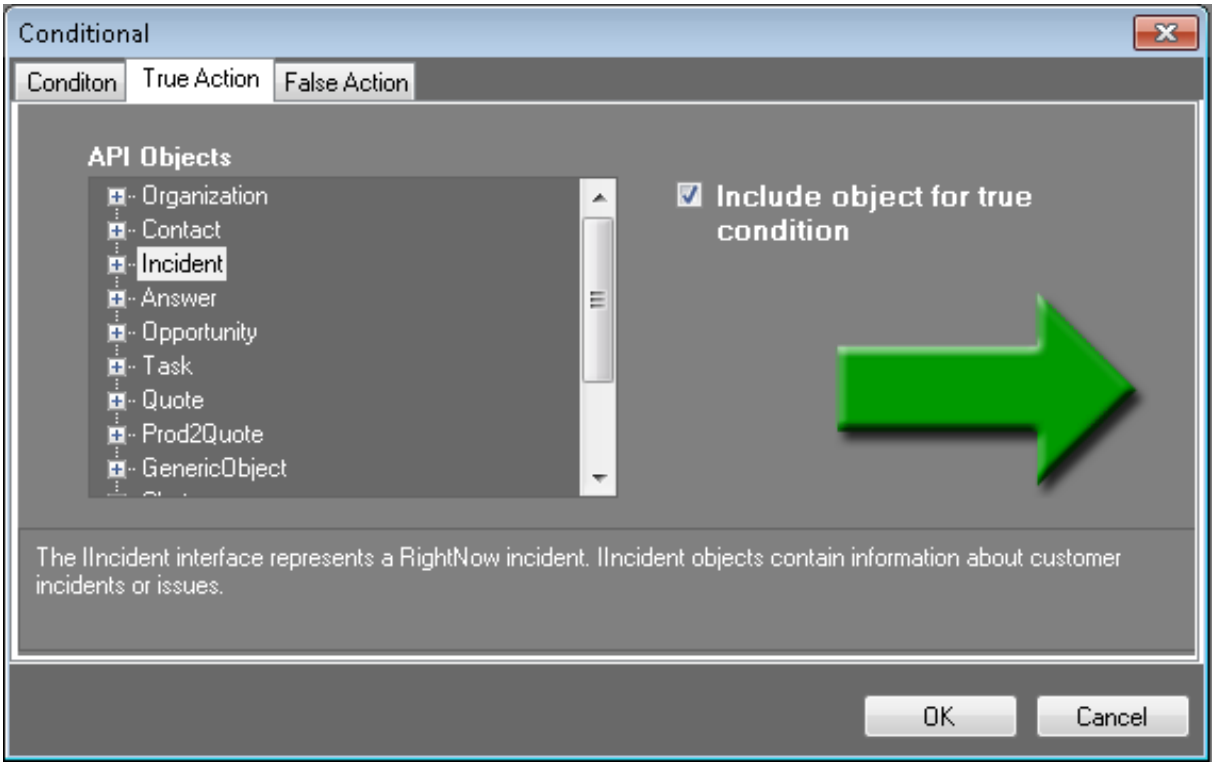



Next is the **Conditional** object.

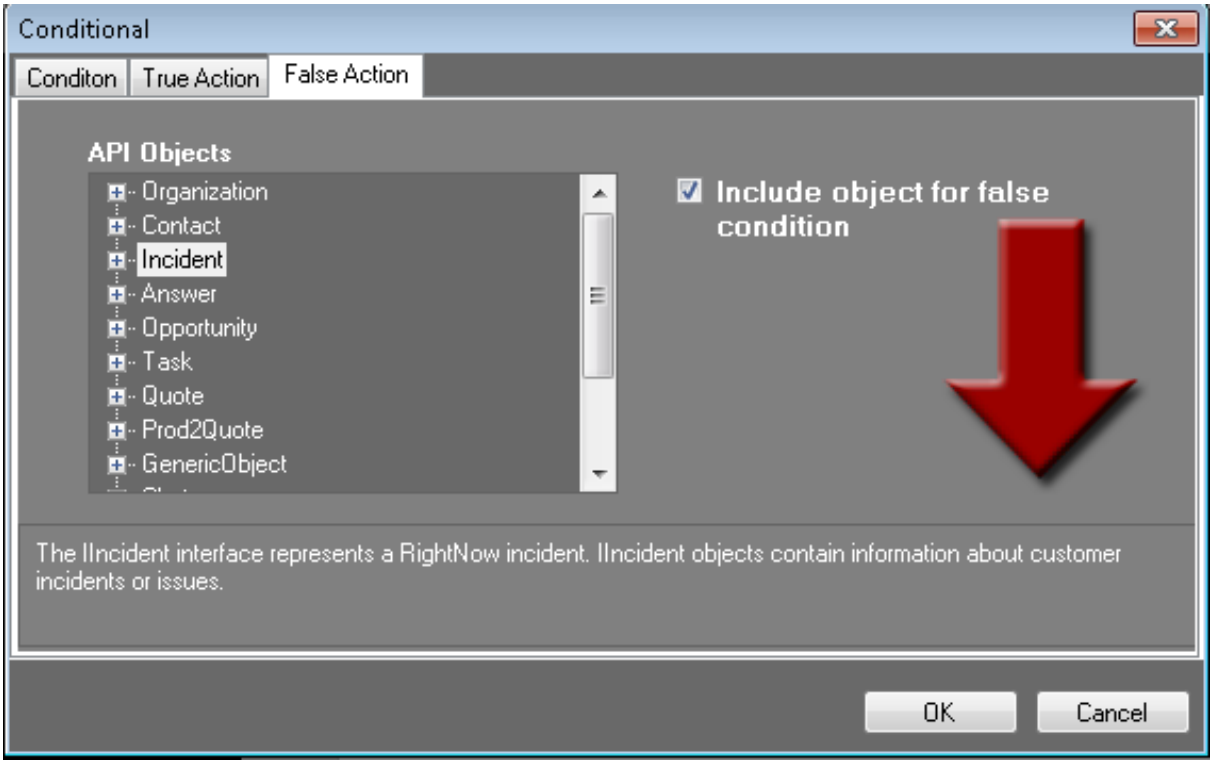


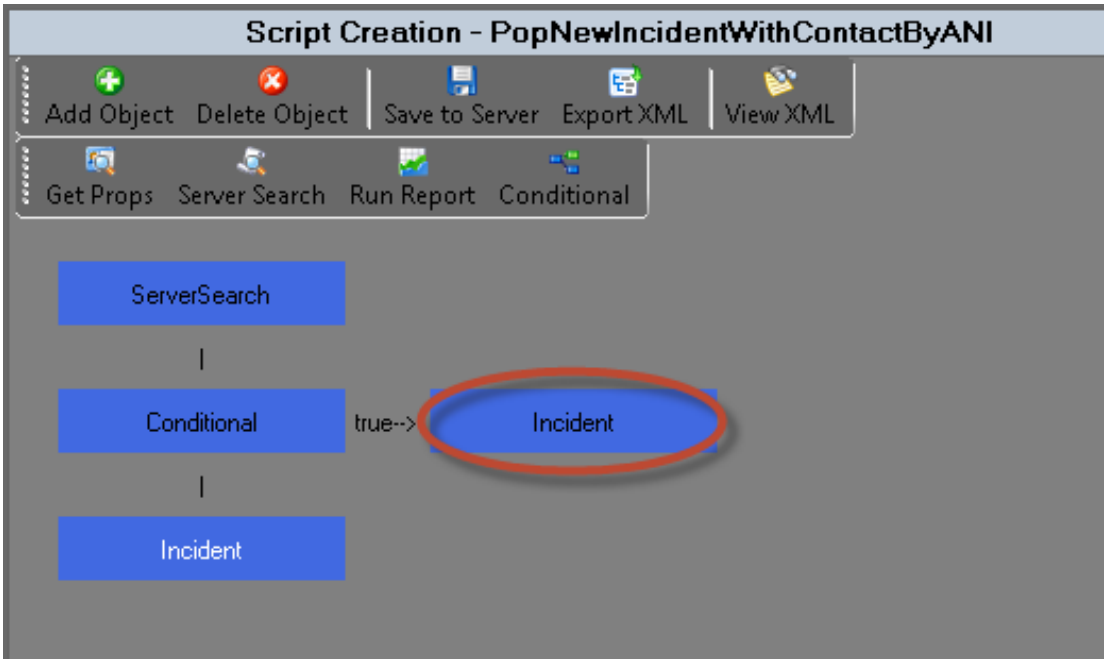

The **Condition** above states that: If the count of **Contact Id** = 1. This means that one record was found.

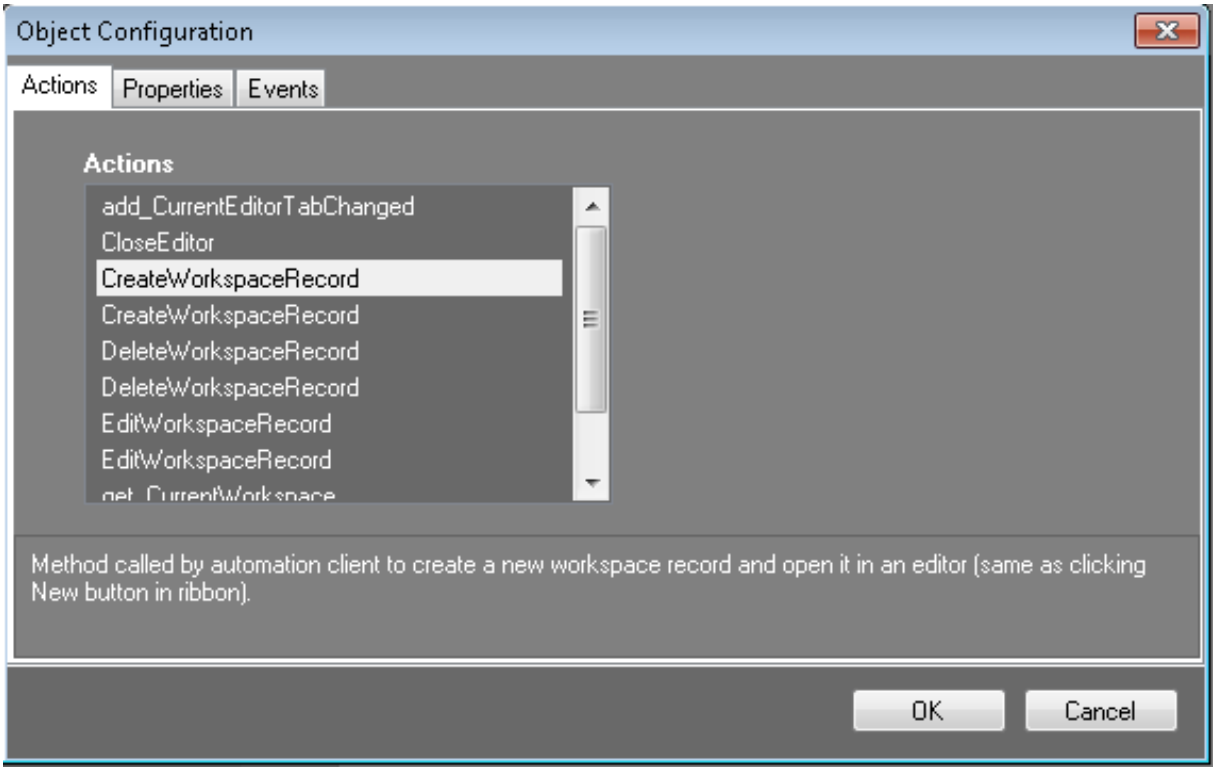



The True Action is: Then pop the **Incident** information.

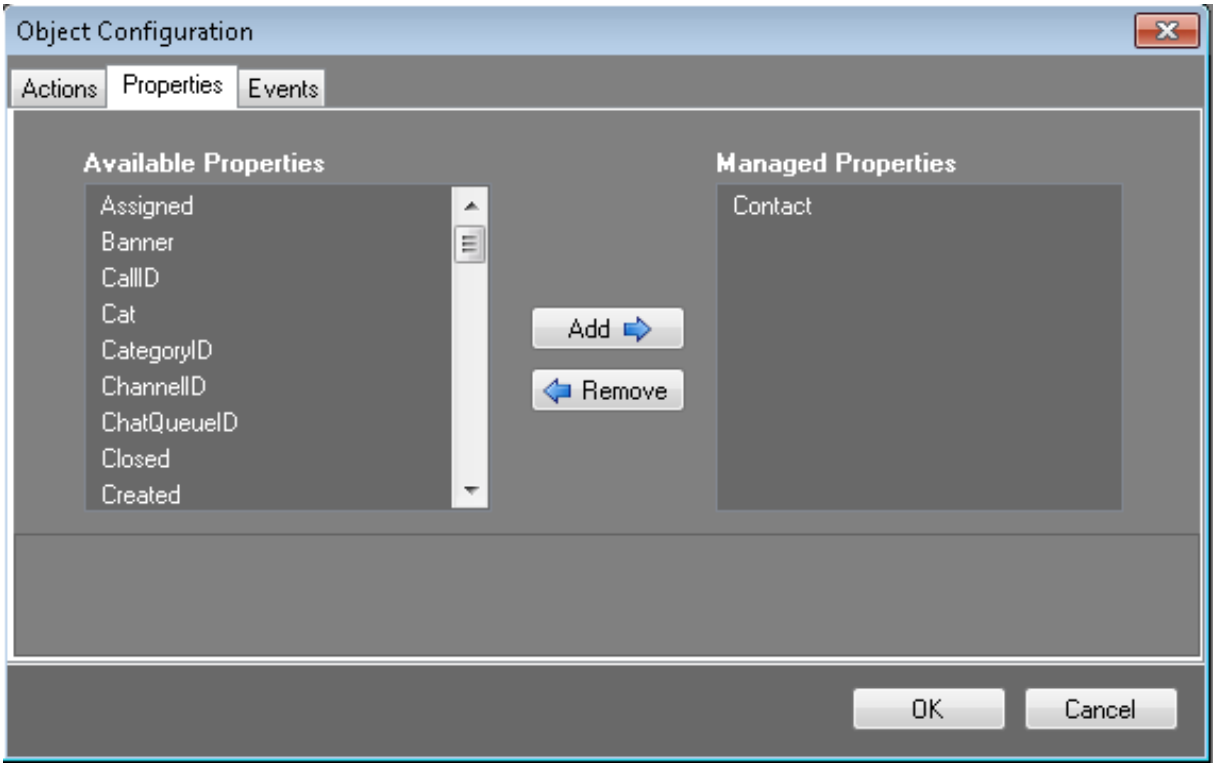



The False Action is: Else pop the **Incident** information. At first glance this seems to be doing the same thing whether True or False, but as you see in the next screen shot there is an **Incident** object if True and a different **Incident** object if False. These finish off the Then and Else of the condition.

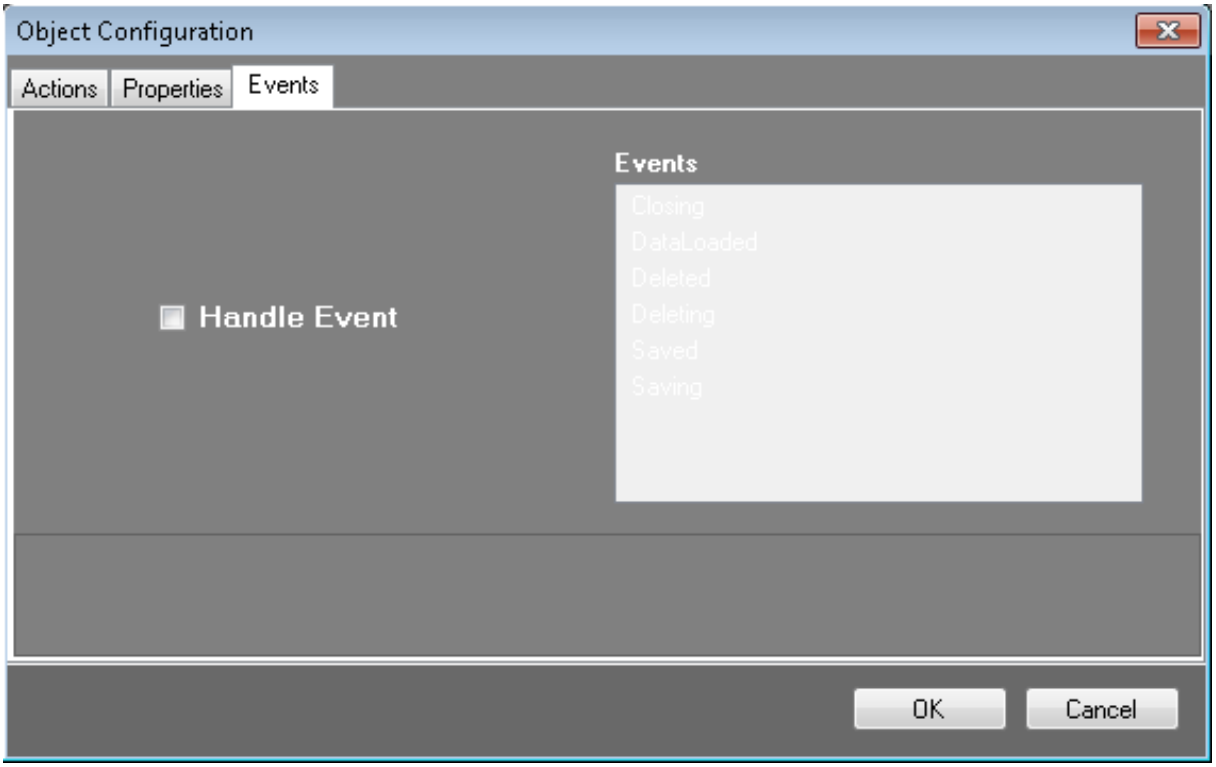



On the **Incident** object of the True side, it is saying pop the record as a New record. It has found the **Contact** record for the person on the phone and it is popping a new **Incident** with that **Contact** information in the create form.

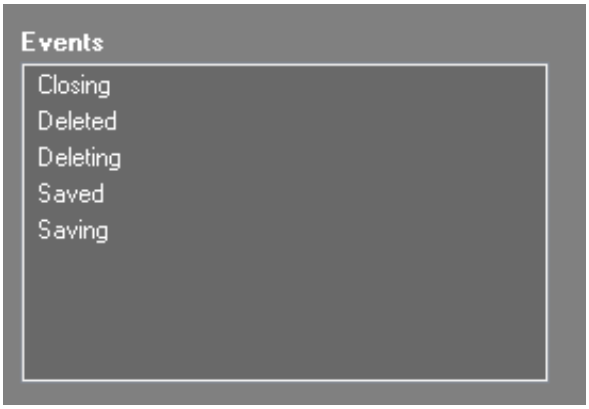


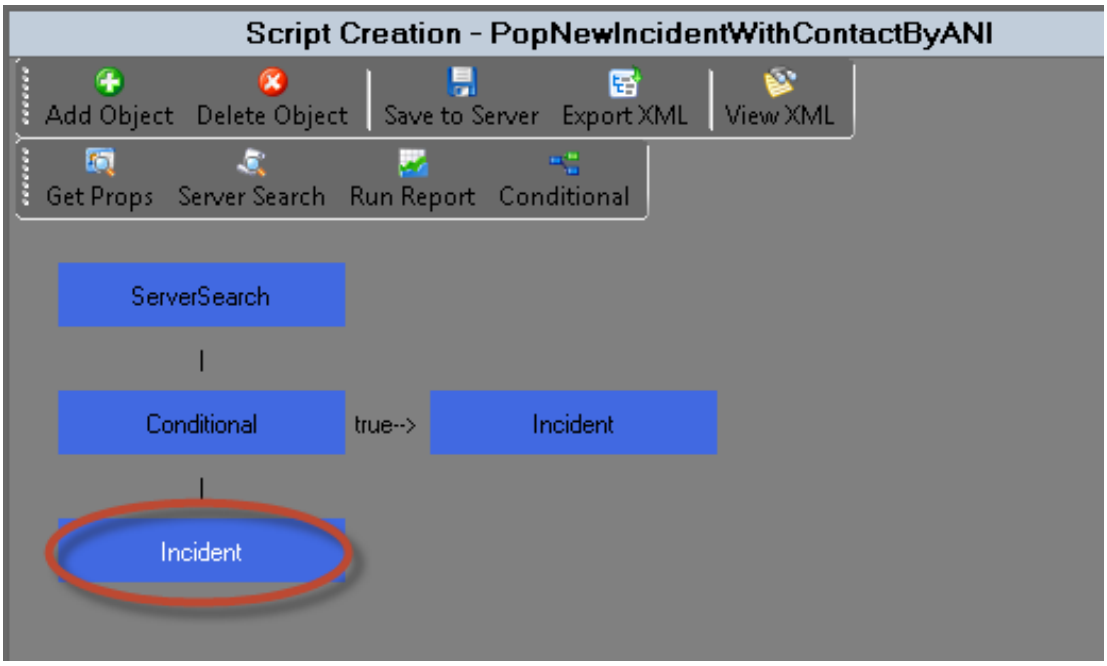

It uses the **Contact** information from the call for the new record.

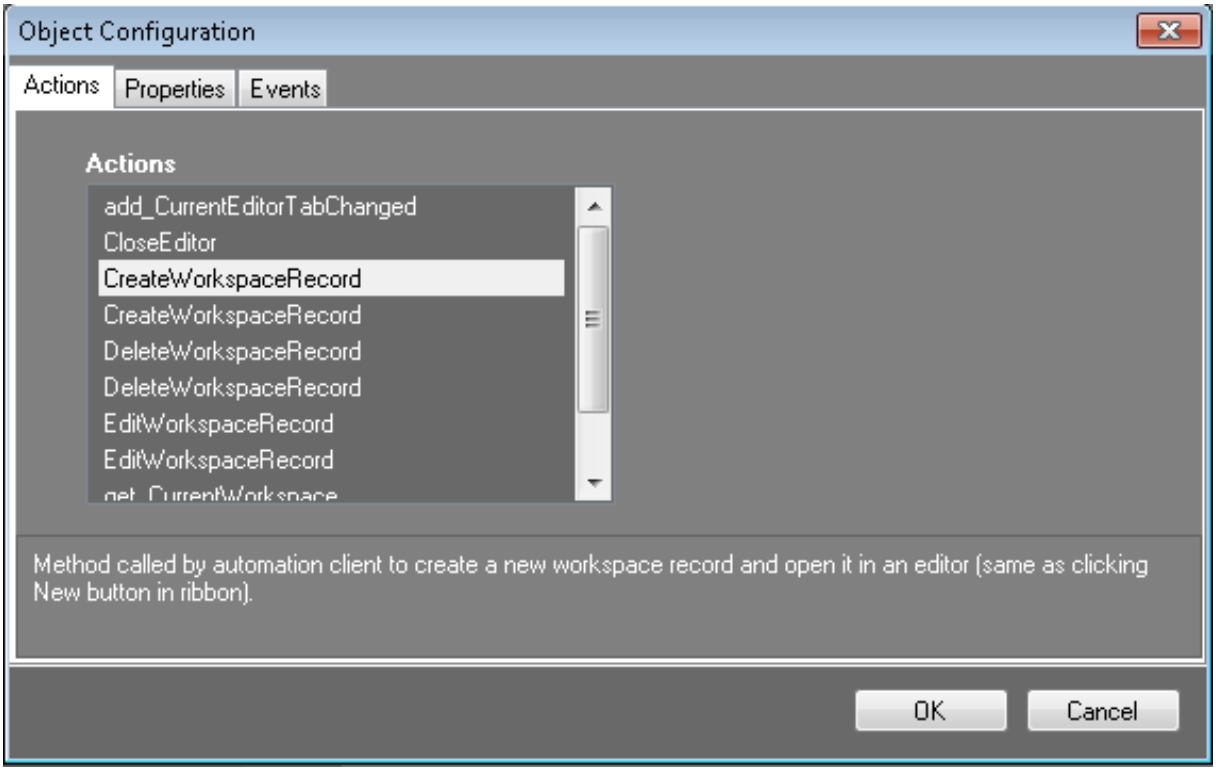



There are no Events chosen.

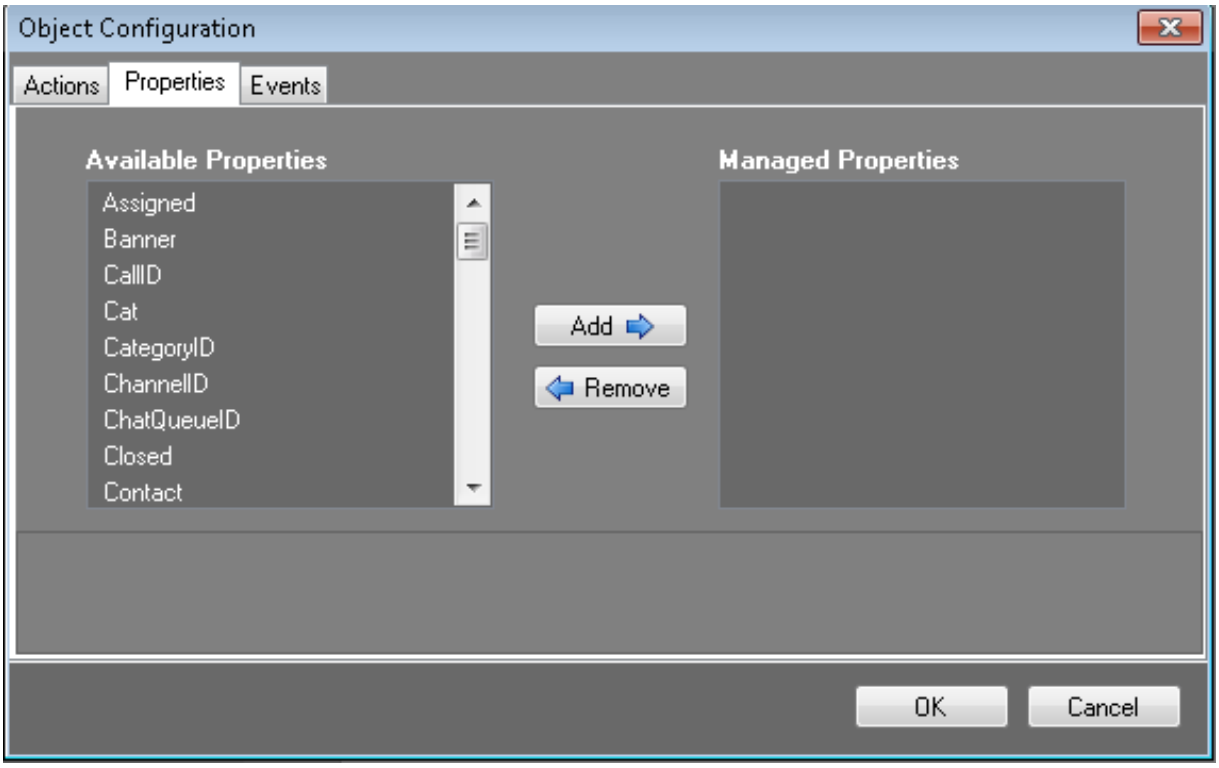
To have an **Event** happen after the action, click the **Handle Event** checkbox and the options become available. You can then choose what happens when the event is fired from Oracle Service Cloud. These events come from the workspace record in Oracle Service Cloud.


Events
Closing
Deleted
Deleting
Saved
Saving

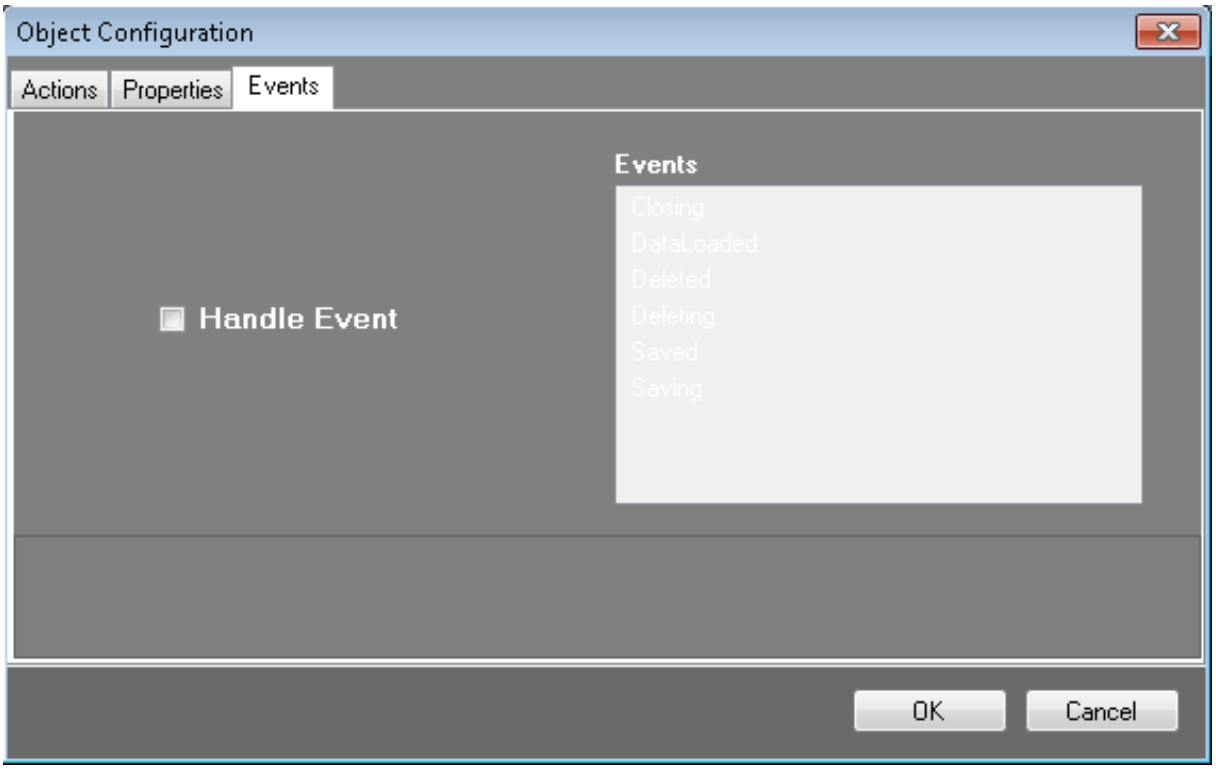

Script Creation - PopNewIncidentWithContactByANI
Add Object
Delete Object
Save to Server
Export XML
View XML
Get Props
Server Search
Run Report
Conditional
ServerSearch
Conditional
true-->
Incident
Incident


Object Configuration
Actions
Properties
Events
Actions
add_CurrentEditorTabChanged
CloseEditor
CreateWorkspaceRecord
CreateWorkspaceRecord
DeleteWorkspaceRecord
DeleteWorkspaceRecord
EditWorkspaceRecord
EditWorkspaceRecord
Method called by automation client to create a new workspace record and open it in an editor (same as clicking New button in ribbon).
OK
Cancel

On the False side it is saying to pop a New record for the **Incident** form. This is because it did not find the contact in the current **Contact** data and needs to create a new **Incident** with new **Contact** information as well.




Since the **Contact** was not found there is nothing being passed.




There are no Events chosen.

Summary of the script:

1. Gets the ANI from the IVR.
2. Looks in the **Contact** data for a match to the caller.
3. If it finds a match it sends that information to the Oracles Service Cloud client and pops a new **Incident** on the screen with the **Contact** information already there.

4. If it doesn't find a match it tells Oracle Service Cloud client to pop a new **Incident** record with no information pre-filled.

## Use existing scripts

Below the **Recent Scripts** is a link to open an **Existing Script**. Exisiting Scripts are XML files on the network. If you created a script from a template as described above, you would open it by clicking the **Existing Script** link and browsing to the location of where you saved that file. Existing scripts are on the network and have not yet been saved to the server.

If scripts are saved as XML, they will show in the **Recent Scripts** box on the main screen. If a script is saved to the server and not saved somewhere else, it will not show in this box.

## Create a new script

You will click the link to **New Script** and build your own script from scratch. In Features of Interaction ScriptAssist for Oracle Service Cloud, each object and toolbar item is described. Using these and the knowledge you can aquire from the templates, you are allowed to create a script from scratch.

You can also use the **File** menu and point to **New** and then click **Screen Pop Script**.

## Open a server script

If you need to modify a script that is already loaded to the server, you will click **File** on the menu and select **Open Server Script**. Once open you can modify it you can save it back to the server, or save it somewhere on the network or your local computer. To save it to the network or the local computer you have to export it to XML.

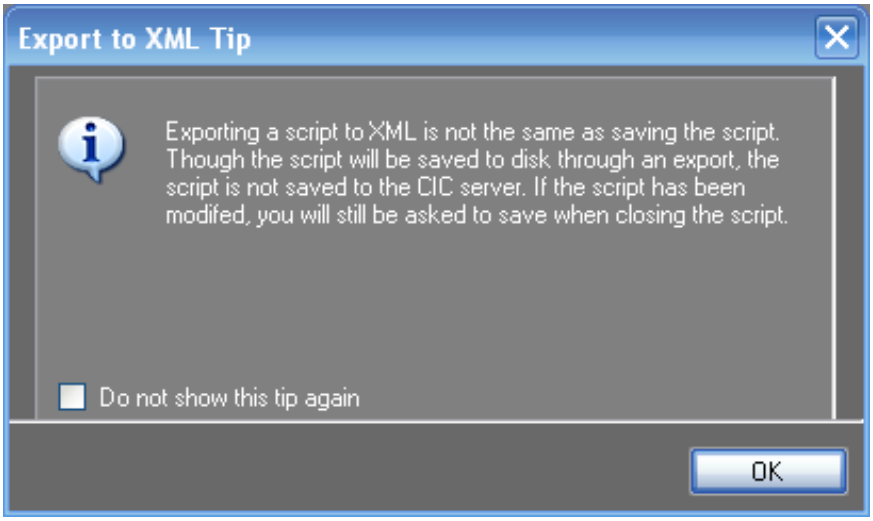
# Use ScriptAssist for Oracle Service Cloud

## Export a script to an XMLfile

If you want to maintain a backup of your scripts, or if you don't currently have an IC server to which you can save a script, then you can export the script locally as XML.

To export a script as an XML file:

1. If needed, open the script in ScriptAssist.
2. Open the **File** menu and click **Export to XML**.
   ScriptAssist displays a message box explaining the process.




3. Click **OK** to close the message box.
4. Browse to the location where you want to save the XML file.
5. Click **Save**.
   ScriptAssist displays a success message.
6. Click **OK** to close the message box.

Your script is now saved as an XML file in the location you selected. When you close the script in ScriptAssist, you will see that the **Recent Scripts** section now lists it.

## Import a script in an XML file

Importing a script that you previously exported as an XML file is no different from opening a script, except that you open it locally instead of remotely on the server.

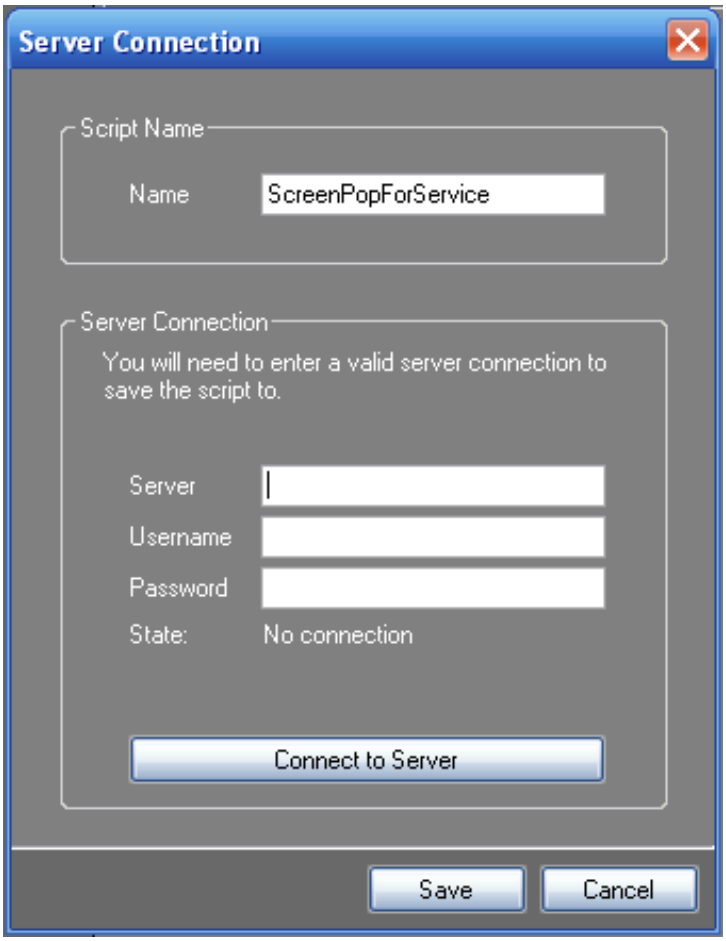
To import a script previously saved in an XML file:

1. Open the **File** menu and click **Import from XML**.
   ScriptAssist displays the **Open** dialog box.
2. Browse to the location of the XML file and open it in ScriptAssist.
   ScriptAssist creates a script based on the information in the XML file.

# Save a script

To save the script currently displayed in the **Script Creation** workspace:

1. Open the **File** menu and click **Save to Server**.
   ScriptAssist prompts you to enter or confirm the server name and credentials.



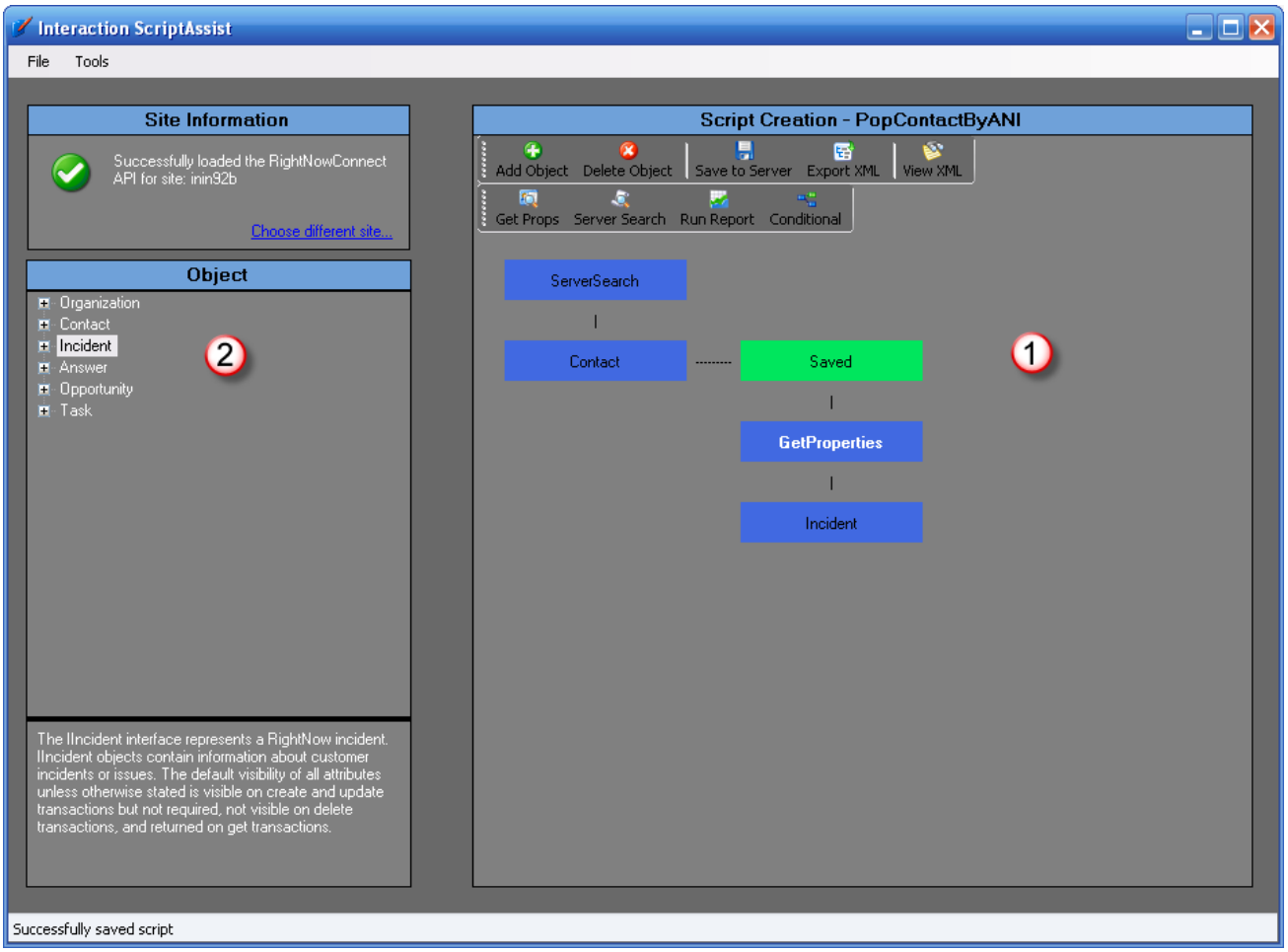

2. In the **Name** text box, type the name to use for the script.
3. If needed, type your server name, username, and password in the corresponding boxes, then click **Connect to Server**.
   If you previously entered these values in the **Preferences** dialog box, then ScriptAssist pre-populates these fields. You do not need to re-enter the same information.
4. Click **Connect to Server**.
   ScriptAssist displays a success message if it connected to the server.
5. Click **Save**.

# Add a node to a script

You can add either a standard node (object) or a special node (operation) to a script.

To add an object node to a script:

1. Create or open the script.
   ScriptAssist displays the **Object** list and **Script Creation** workspace.




2. In the **Script Creation** workspace (1), click the node *after which* you want to insert the new node.
   New nodes will be inserted after the selected node. If you are inserting the first node in a script, skip this step.
3. In the **Object** list (2), double-click the type of object node you want to insert.
   ScriptAssist inserts the new node immediately after the node you clicked in Step 2.

   > **Tip:**
   > You can also add a node by clicking the **Add Object** button in the toolbar.

4. Configure the new node's properties, actions, and events as needed.

For more information, see Configure an object node.

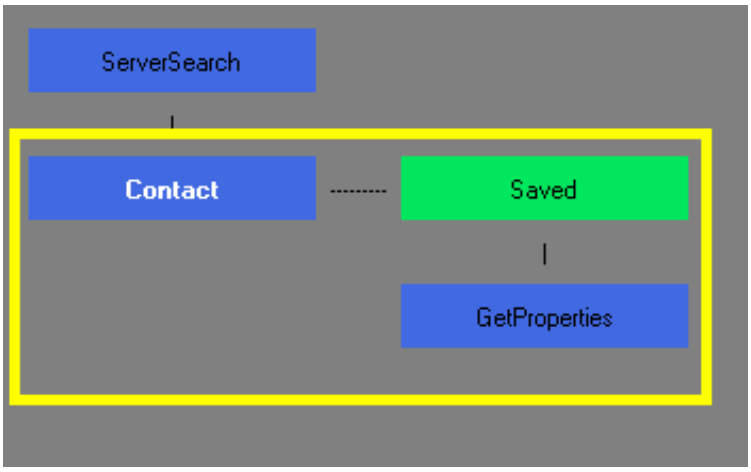
> **Note:**
> As shown above (1, contact node), it is *possible* to add more than one node after another node. However, the script executes from the top left of the diagram to the bottom right. Script branching and multithreading are not supported in the current release of ScriptAssist. In the script shown above, the **Server Search** would first execute followed by the **Contact** node. After the contact is saved, the script would execute a **Get Properties** node followed by an **Incident** node.

## Delete a node from a script

To delete a node:

1. In the **Script Creation** workspace, click the node to delete.
2. In the script toolbar, click the **Delete Object** button.
   ScriptAssist prompts you to confirm the deletion.
3. Click **Yes** in the dialog box.

ScriptAssist deletes the selected node *and all of its child nodes* in the script.

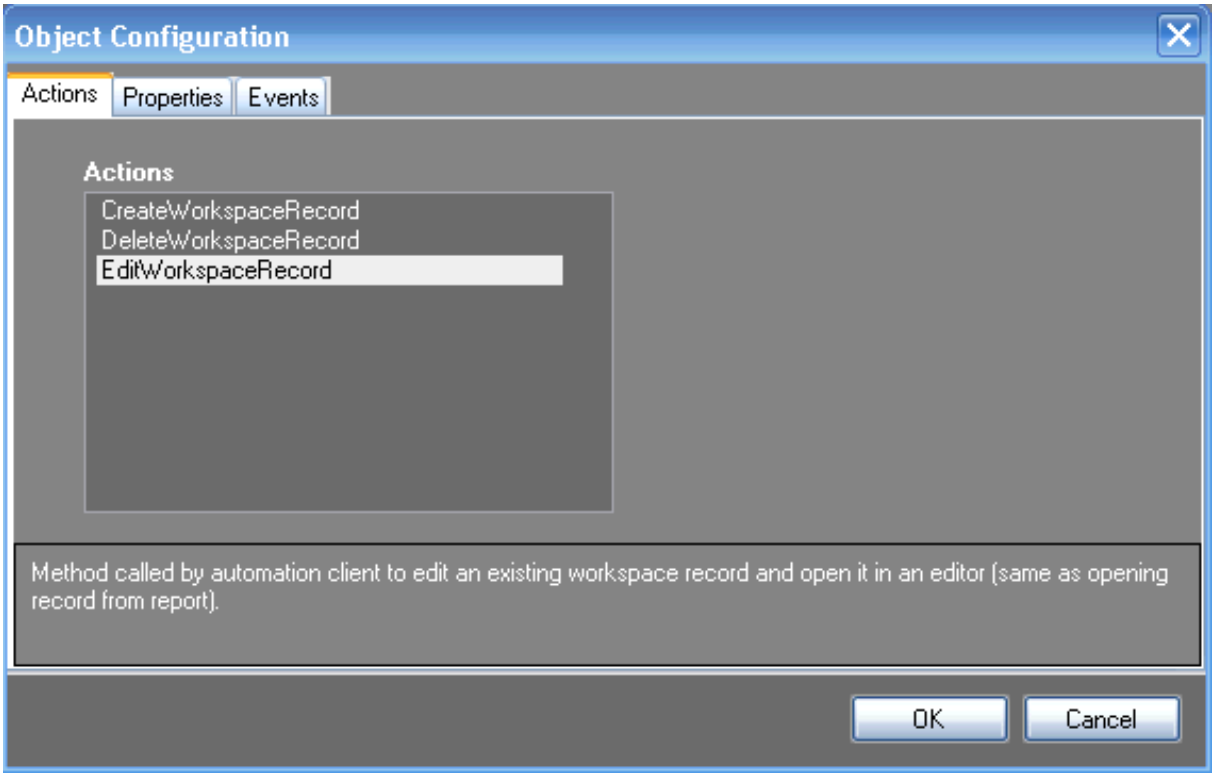



## Configure an object node

Configure an object node by setting its action and its properties, and selecting an event for it to handle. Only the first, setting an action, is always required.

To configure an object node:

1. In the **Script Creation** workspace, double-click an object node.
   ScriptAssist displays the **Object Configuration** dialog box with the **Actions** tab on top.

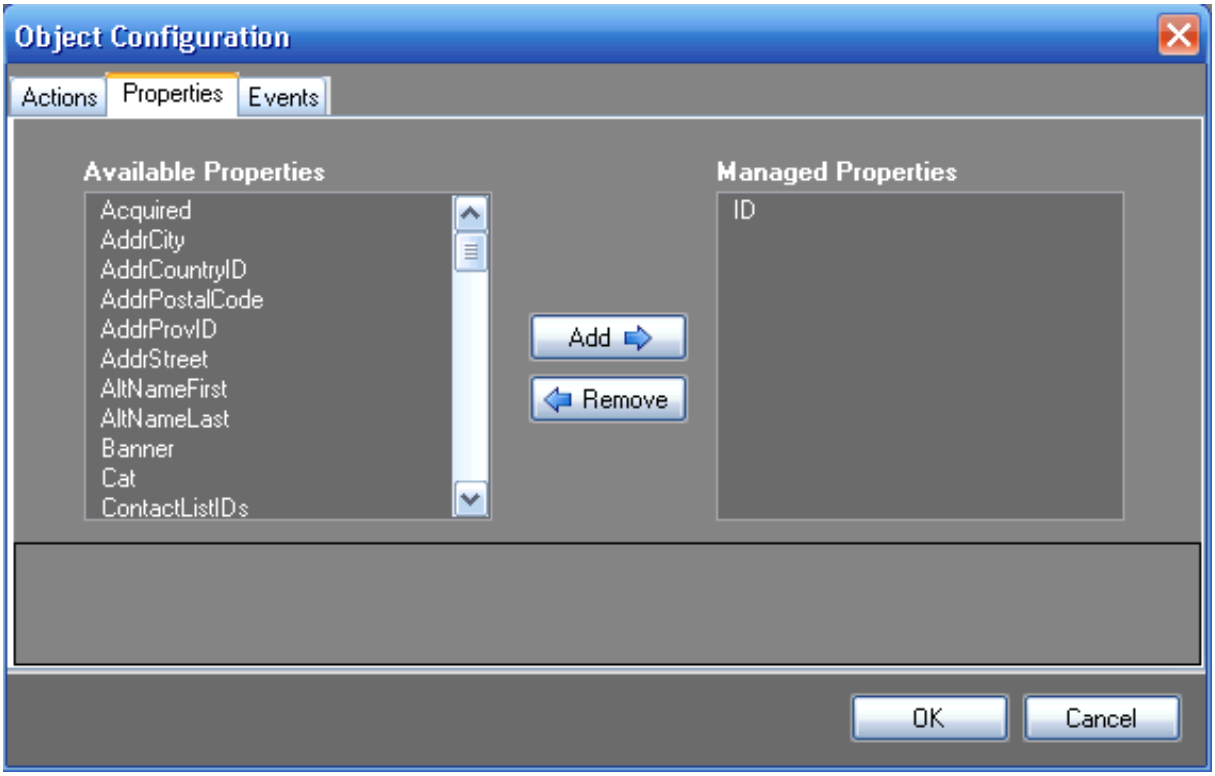



2. In the **Actions** list, click the desired action.

- **CreateWorkspaceRecord** to cause a screen pop of a new workspace record.
- **DeleteWorkspaceRecord** to delete an existing workspace record on the Oracle Service Cloud server.
- **EditWorkspaceRecord** to cause a screen pop of an existing **workspace** record.

> **Note:**
> The area at the bottom of the dialog displays a brief explanation of the selected action.

3. To manage a property, click the **Properties** tab.




You manage a property in order to apply some value (whether it is a static value or a token) to the property. Use this if you want to pre-populate some fields on that workspace record.

To set a value on a property for the selected object, click the property in the **Available Properties** list and then click the **Add** button.

ScriptAssist displays the **Custom Property Configuration** dialog box.

a. Use this dialog to set values on the managed property.

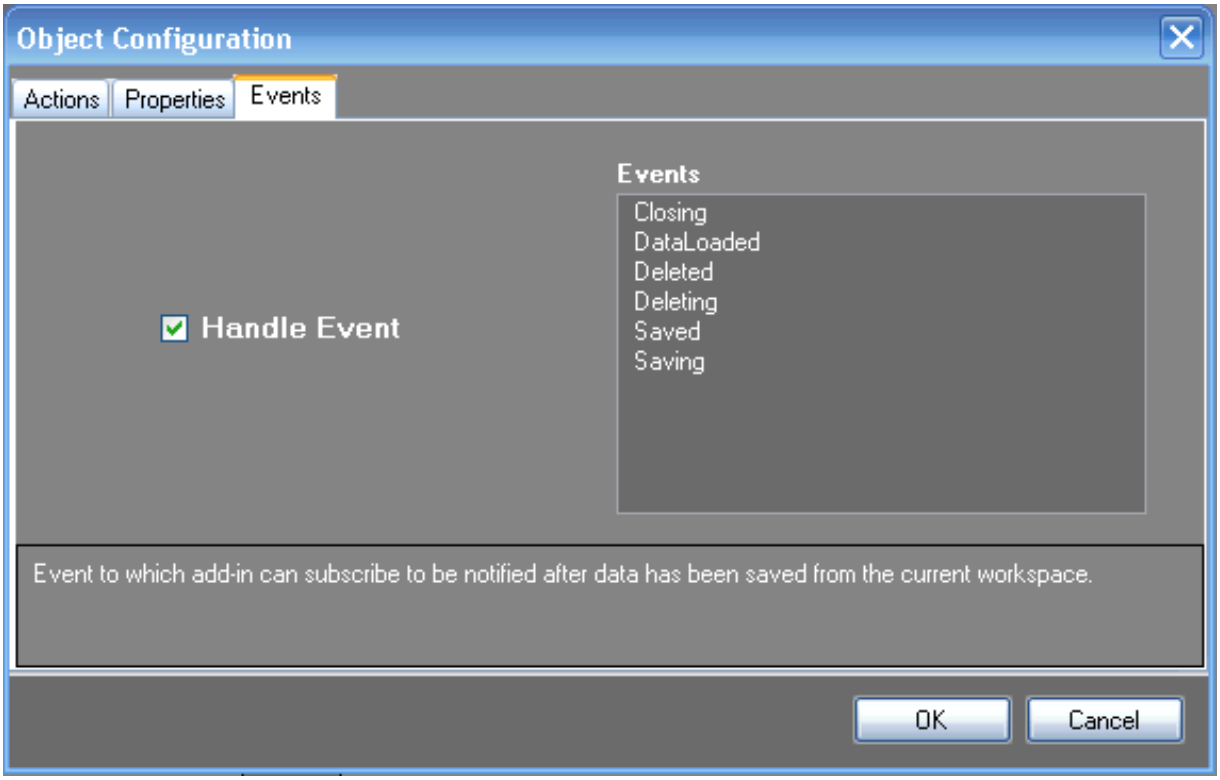
   For more information about customizing properties, see Understanding and configuring node properties.

b. Click **OK** to close the **Custom Property Configuration** dialog box.

c. Manage more properties as needed.

The properties of each node type are defined by Oracle Service Cloud and are fully documented in your Oracle Service Cloud Desktop Add-Ins documentation. When you select a property, the bottom area of the dialog shows the Oracle Service Cloud Desktop Add-Ins documentation for that property.

4. To select an event to handle, click the **Events** tab. For more information, see Events that objects can handle.




To make the node handle a particular event, select the **Handle Event** check box and click the desired event in the **Events** list.

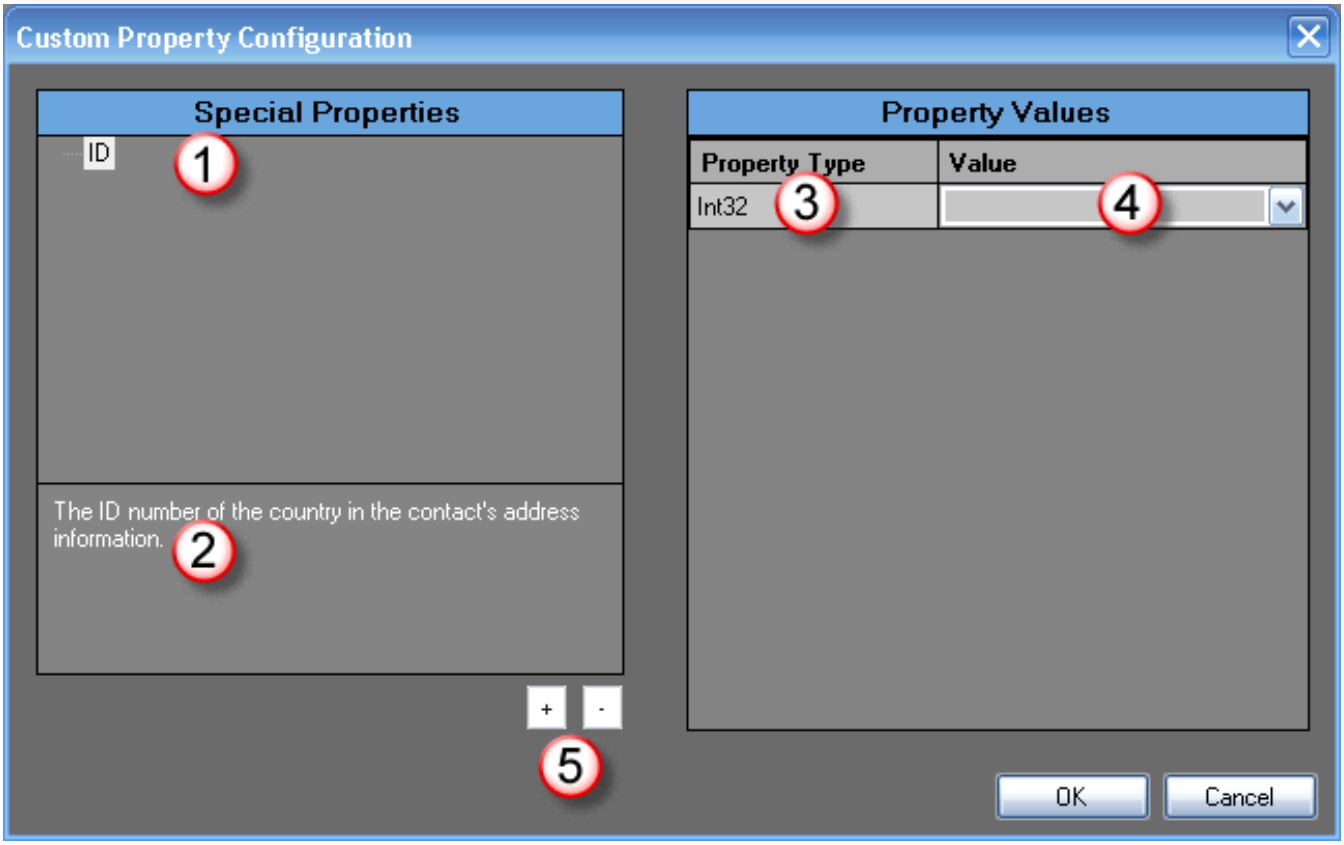
# Understanding and configuring node properties

Each node in your script has properties that can store values of appropriate types. Properties allow you to apply values to the nodes in your script.

You choose properties to which to apply values (ScriptAssist calls this "managing" the properties) on a node's **Properties** tab. For more information, see Configure an object node.

You retrieve property values to use elsewhere by using a **Get Properties** node. For more information, see Use a Get Properties node.

Whenever you manage a property, ScriptAssist displays the **Custom Property Configuration** dialog box.




The main parts of the **Custom Property Configuration** dialog are:

- Property name (1): This is the name of the property to which you can apply a value or token.
- Property description (2): This is a description of the selected property. It is taken from the Oracle Service Cloud Desktop Add-Ins documentation.
- Property Type (3): This is the data type of the selected property.
- Value (4): This is a combo box in which you can type or select the value of the property, such as `{IVR_Replacement_Token}`.
- Add/remove buttons (5): An incident can have zero or more contacts. When you manage the contact property, you can add one or more contact objects using the (+) button, and you can remove contacts from the list with the (–) button.

## Events that objects can handle

Handling events is a way to pause script engine execution. For example, if you create a script with three object nodes in succession, then they would produce three nearly simultaneous screen pops. By handling an event on each object, you can synchronize the screen pops so that they occur separately.

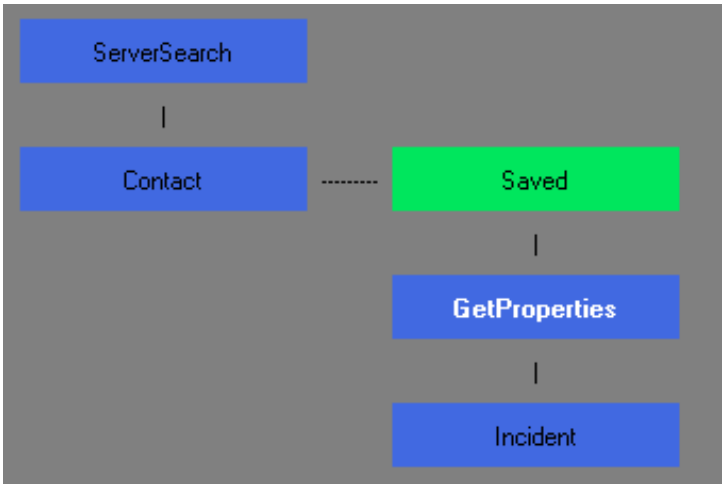
Use an object node's **Events** tab to choose an event for the object to handle. Handled event types are:

- Closing: When the user initiates closing a record, the script moves on to the next node.
- DataLoaded: When data has finished loading for a record, the script moves on to the next node.
- Deleted: When a record is deleted, the script moves on to the next node.
- Deleting: When the user initiates a delete on a record, the script moves on to the next node.
- Saved: When a record has been saved, the script moves on to the next node.
- Saving: When the user initiates a save on a record, the script moves on to the next node.

When you edit an object's properties, you can select one of these events on the **Events** tab of the **Object Configuration** dialog box. For more information, see Configure an object node.

When you tell an object to handle an event, the script automatically watches for that event. When the event occurs, the script takes it as a signal to go on to the next node.

For example, you might tell a **Contact** node to handle the **Saved** event. This would place a **Saved** node immediately after the **Contact** node.




When the user saves the contact information in response to the screen pop, then the **Contact** node detects the event and the script moves through the **Saved** event node on to the next node in the script. In the above example, the next node is a **GetProperties** node that makes a copy of whatever values the user saved and can make those values available to the next node or any node after that.

## Special node types (for script operations)

The special node types in ScriptAssist represent script operations. These node types are **Server Search**, **Get Properties**, **Run Report**, and **Conditional**.

For more information, see the following:

- Use a Server Search node
- Use a Get Properties node
- Use a Run Report node

### Use a Server Search node

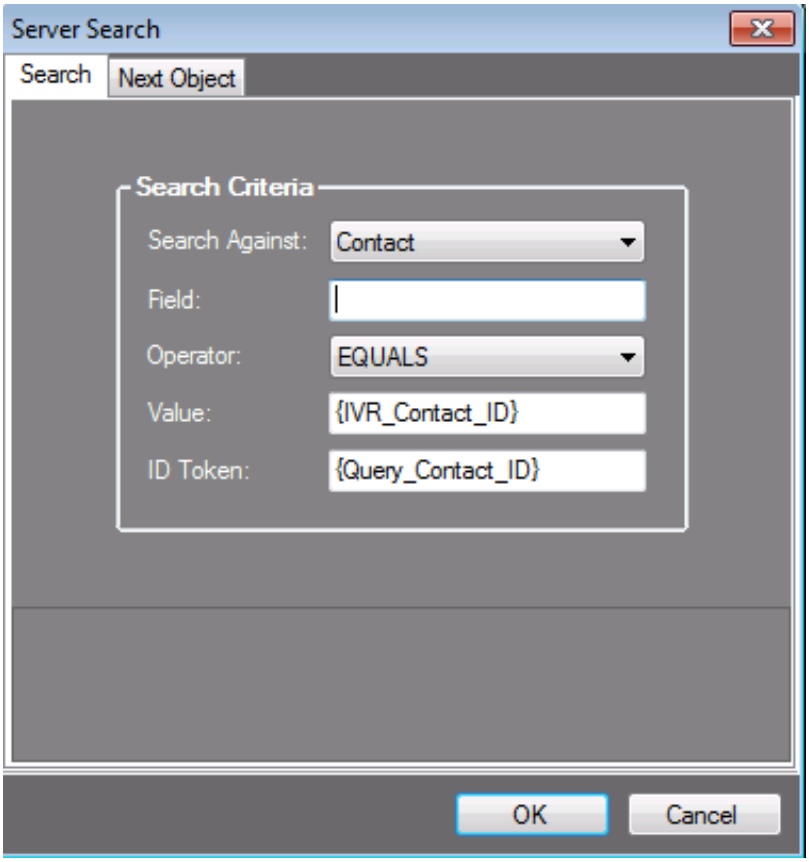
Use the **Server Search** node to make the script engine search the Oracle Service Cloud server for specific records.

To configure a **Server Search** node:

1. In the script toolbar, click the **Server Search** 

button.
   ScriptAssist displays the **Server Search** dialog box.

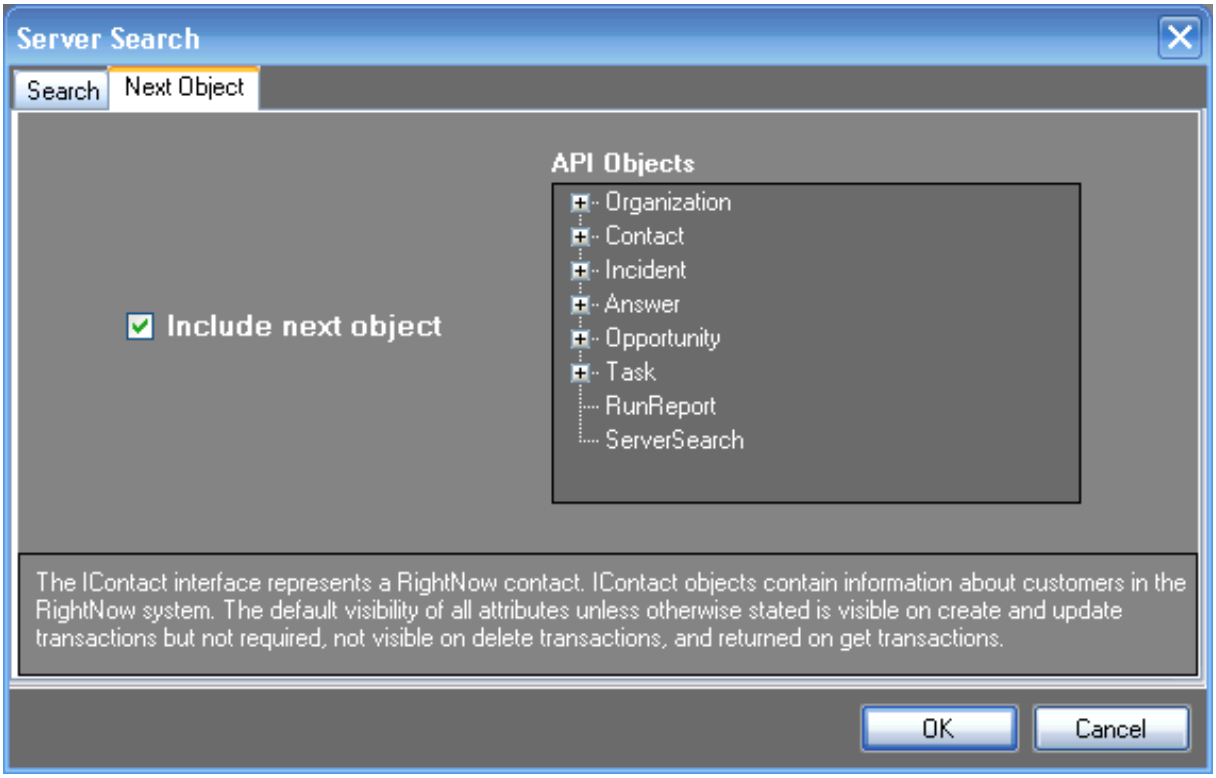
   **Note:**
   The dialog box will not be populated when it appears. Its fields will be empty.




2. In the **Search Criteria** area:
   a. Expand the **Search Against** list box and select the object type for which to search.
   b. In the **Field** text box, add any readable field that is relevant to the object type you selected in the **Search Against** list box. For a list of readable fields, see Oracle Service Cloud Connect Web Services for SOAP (Connect Web Services) found on the Oracle Service Cloud website.
   c. Expand the **Operator** list box and select the comparison operator to use (equals, greater than, etc.).
   d. The **Value** field is populated for you based on your **Search Against** field selection. By default, it is populated with an IVR Replacement Token. The token may be replaced with a static value, or may be renamed. If the token is renamed, it must start and end with a brace `{}`.
   e. The **ID Token** field is populated based on your **Search Against** field selection. By default it is populated with a Query Replacement token. The field may be modified, but will always contain a Query Replacement Token (it cannot contain a static value; the field won't let you add a static value). Query Replacement Tokens always start and end with braces `{}`.
3. If desired, select an object node to insert immediately after the **Server Search**:

   **Note:**
   The script executes the next object node immediately after the server search completes.

   a. Display the **Next Object** tab.




b. Select the **Include Next Object** check box.
c. In the **API Objects** list, click the object type to insert.

4. Click **OK**.

## Search results

You can set your script to pop workspaces of search results. Due to the slow responsiveness of the Oracle Service Cloud UI, ScriptAssist only pops a maximum of 20 workspaces.

> **Note:**
> With larger numbers of workspaces, the Oracle Service Cloud UI may be slow or temporarily unresponsive. Until the UI is done loading, use a softphone or physical phone.

If a search returns more than 20 results, no workspaces pop. Instead, the following error appears in the logs:

```
WorkspaceScriptActionCommand.Execute : There are {x} records trying to pop and this is too many.
Please modify the script and run a report if more than 20 records are found.
```

To see more than 20 search results:

1. Modify the script
2. Run a report

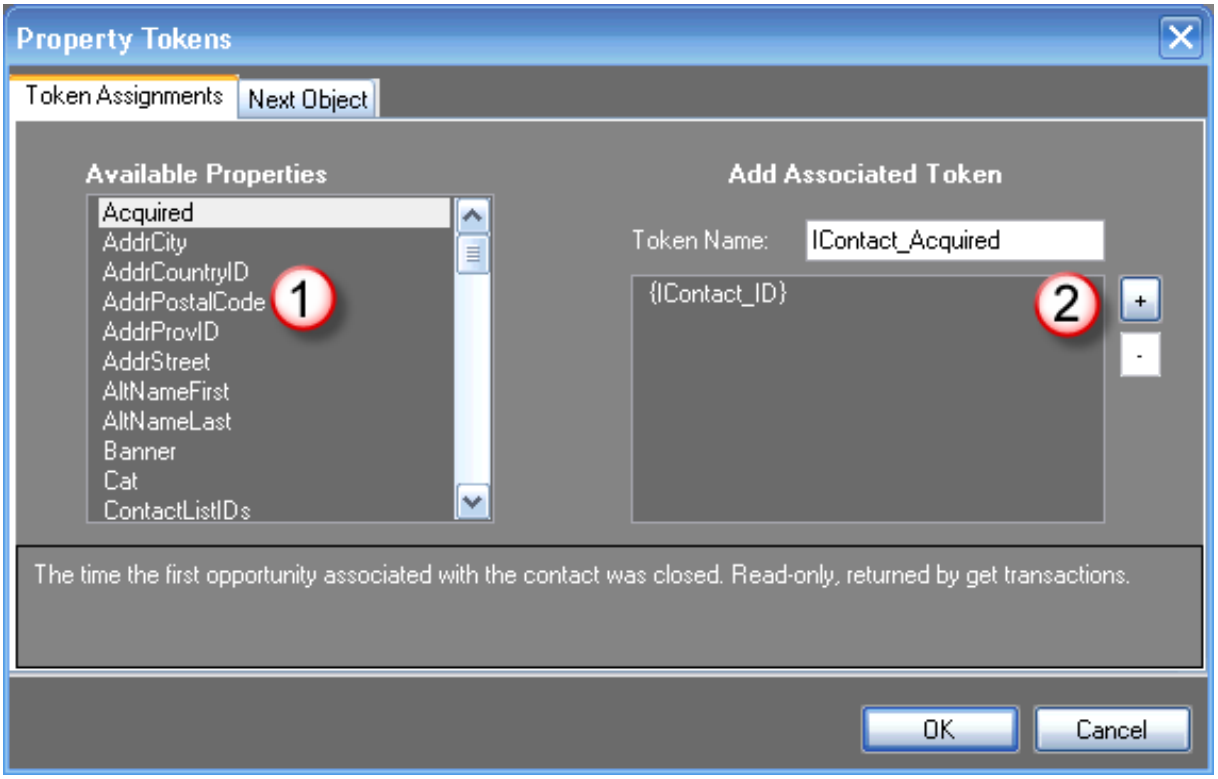
## Use a Get Properties node

Use the **Get Properties** after a **Saved** event to get the values on the object previously saved. You can pass those values from the **Get Properties** node to any subsequent node in your script. Values are not committed to an object until the object is saved. If you want to reference a property on an object later in the script, then there needs to be a way to capture values on that object *after* they have been committed. The **Get Properties** node accomplishes this for you.

To configure a **Get Properties** node:

1. On the script toolbar, click the **Get Properties** 

button.
ScriptAssist displays the Property Tokens dialog box.




2. In the **Available Properties** list (1), click each property whose value you want to retrieve for use later in the script, then click the Plus (+) button (2) next to the **Add Associated Token** list.
   This creates a token which is a unique, meaningful name for that property's value. The token can be used later in the script to inject the property value on some other object. To rename a property, edit the name in the **Token Name** box, then click the Plus (+) button as you normally would.
   To remove a token that you previously added, select it in the **Add Associated Token** list, then click the Minus (–) button. To rename a token, select it in the **Associated Tokens** list, edit its name in the **Token Name** text box, and click the Plus (+) button.
3. On the **Next Object** tab, select the next object type for the script. Then click **OK**.

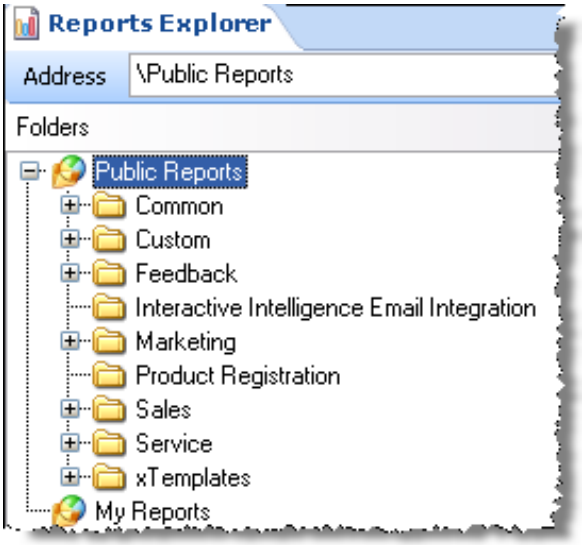
## Use a Run Report node

Use the **Run Report** node to make the script run one of Oracle Service Cloud's predefined reports. Reports on the Oracle Service Cloud server might or might not require additional filters to run. The **Run Report** node lets you to specify any required filters to apply to the report.

Before you use a **Run Report** node, you must know the Report ID (AcID) of the report you want to run. You must also know the names of the filters that are pre-defined in the report. Both pieces of information are in the report definition.

To obtain the report ID and filter names:

1. Start the Oracle Service Cloud client.
2. In the **Analytics** section, double-click **Reports Explorer**.
3. In the **Reports** folders, locate the report you want to run.



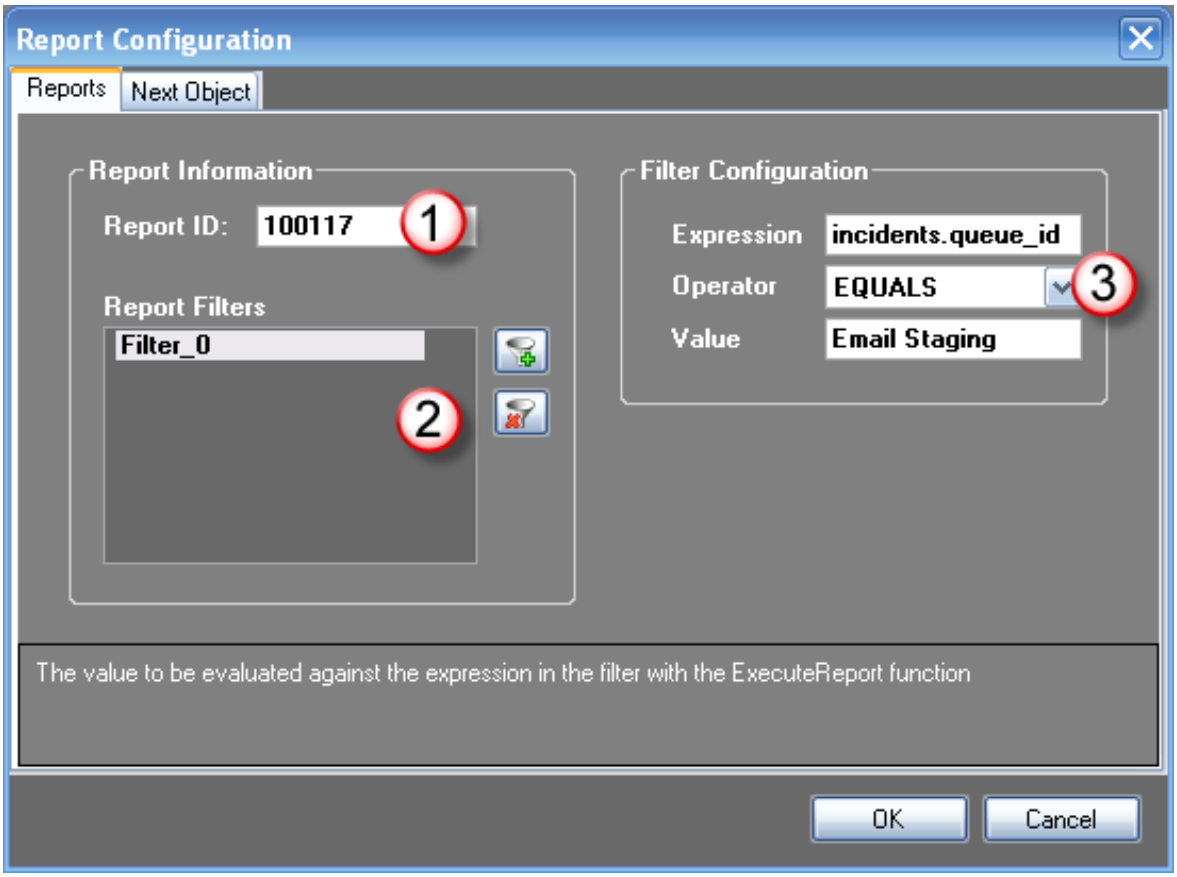

4. Right-click the report and click **View Definition**.
5. Note the information you need:
   a. From the **AcId** line, note the report ID.
   b. From the **Filters** list, note the filter names.

   The filter name is the part of the filter expression that occurs to the left of the equals sign.
6. Close the **Report Definition** window and exit the Oracle Service Cloud client.

To configure a **Run Report** node:

1. On the script toolbar, click the **Run Report** 

button.
   ScriptAssist displays the Report Configuration dialog box.




2. In the **Report ID** box (1), type the ID number of the report that the script should run.

You obtain this ID from the **Analytics** section of the Oracle Service Cloud client.

3. If the report you selected requires a filter, click the **Add Filter**  button to add a filter to the **Report Filters** list (2).

   **Note:**
   Before adding a filter in the **Report Configuration** window, first add a filter of the configuration to the report in Oracle Service Cloud. This filter then appears in a list of filters associated with the report in Oracle Service Cloud.

   To delete a filter from the list, select it and click the **Delete Filter**  button.
4. In the **Filter Configuration** area (3), enter the expression to match, the comparison operator, and the value to match in the corresponding boxes. The expression should match a field found in the report's filter definition.
5. If needed, click the **Next Object** tab and select a next object for the script.
6. Click **OK**.

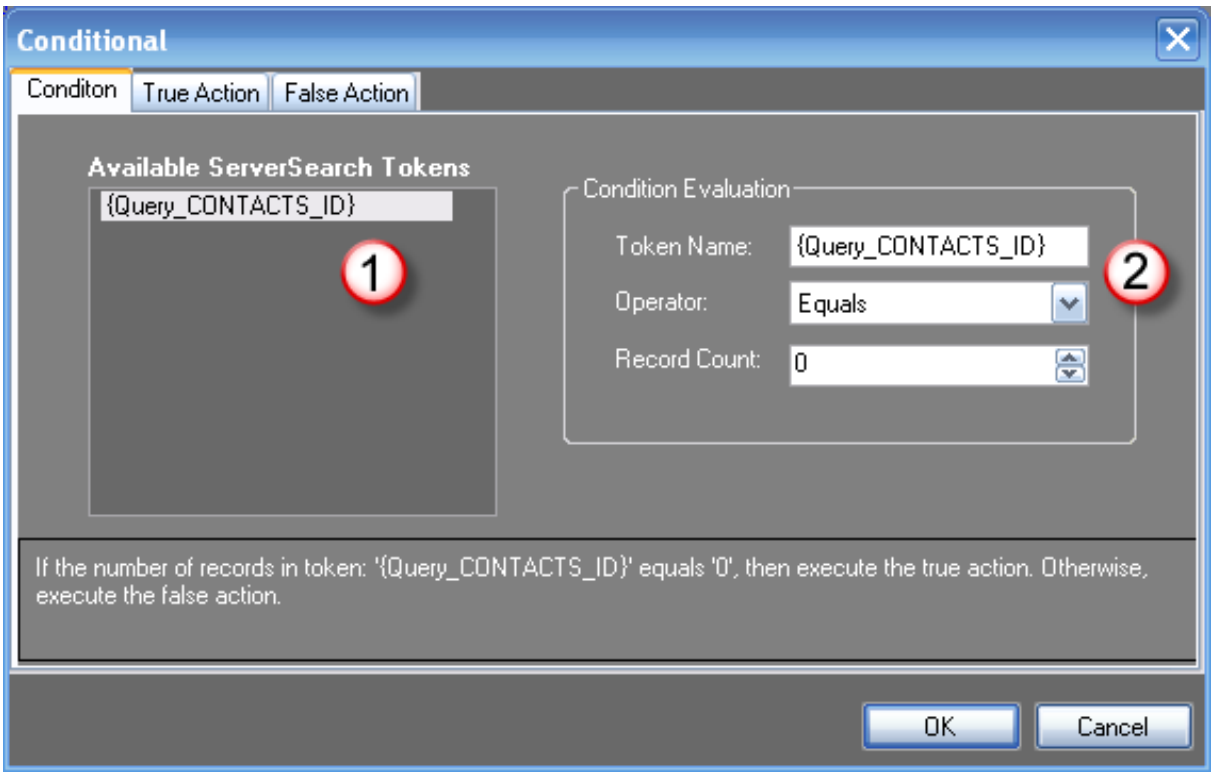
## Use a Conditional node

Use a **Conditional** node after a **Server Search** node to include one API object or another based on whether or not a condition is true.

To configure a conditional node:

1. On the script toolbar, click the **Conditional** 

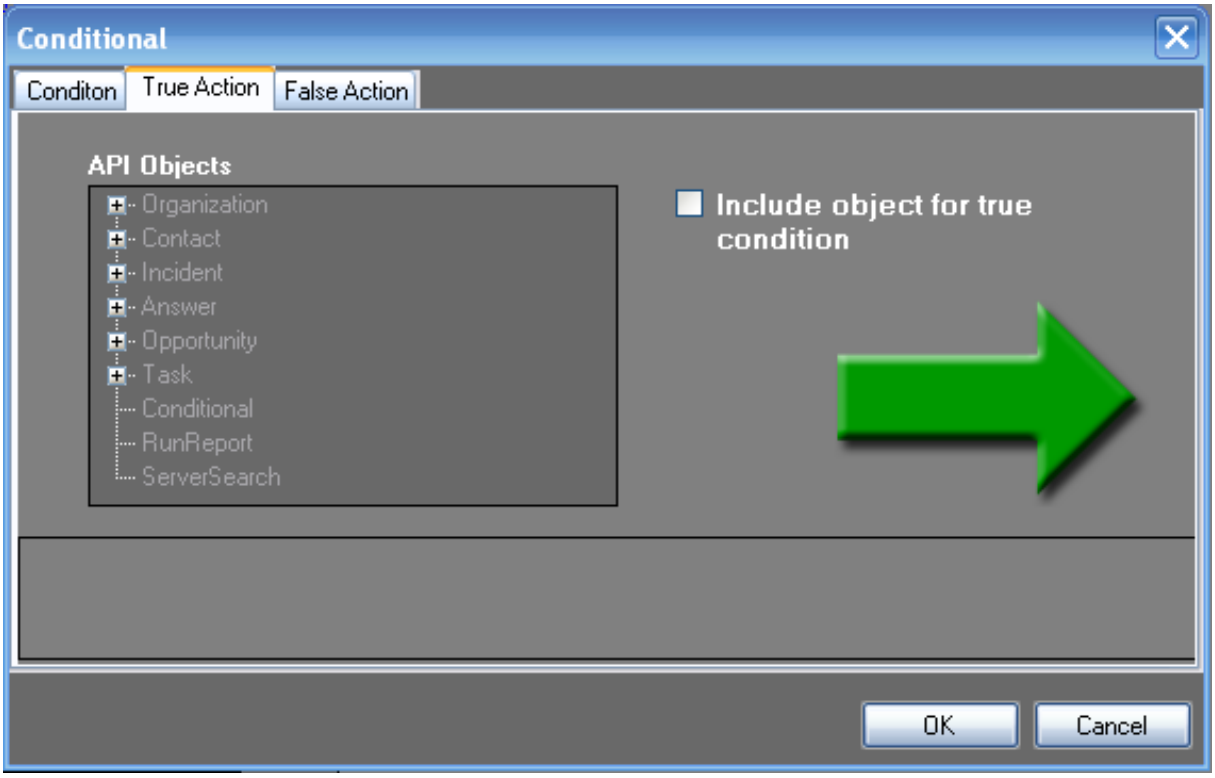
button.
   ScriptAssist displays the **Conditional** dialog box.




2. In the **Available ServerSearch Tokens** list (1), select the token name to use in the conditional expression.
3. In the **Condition Evaluation** group box (2), construct the expression:
   a. The **Token Name** text box shows the token that will be evaluated for the condition.
   b. In the **Operator** combo box, select the desired operator for the expression.
   c. In the **Record Count** box, use the up and down buttons to enter the value for the expression.

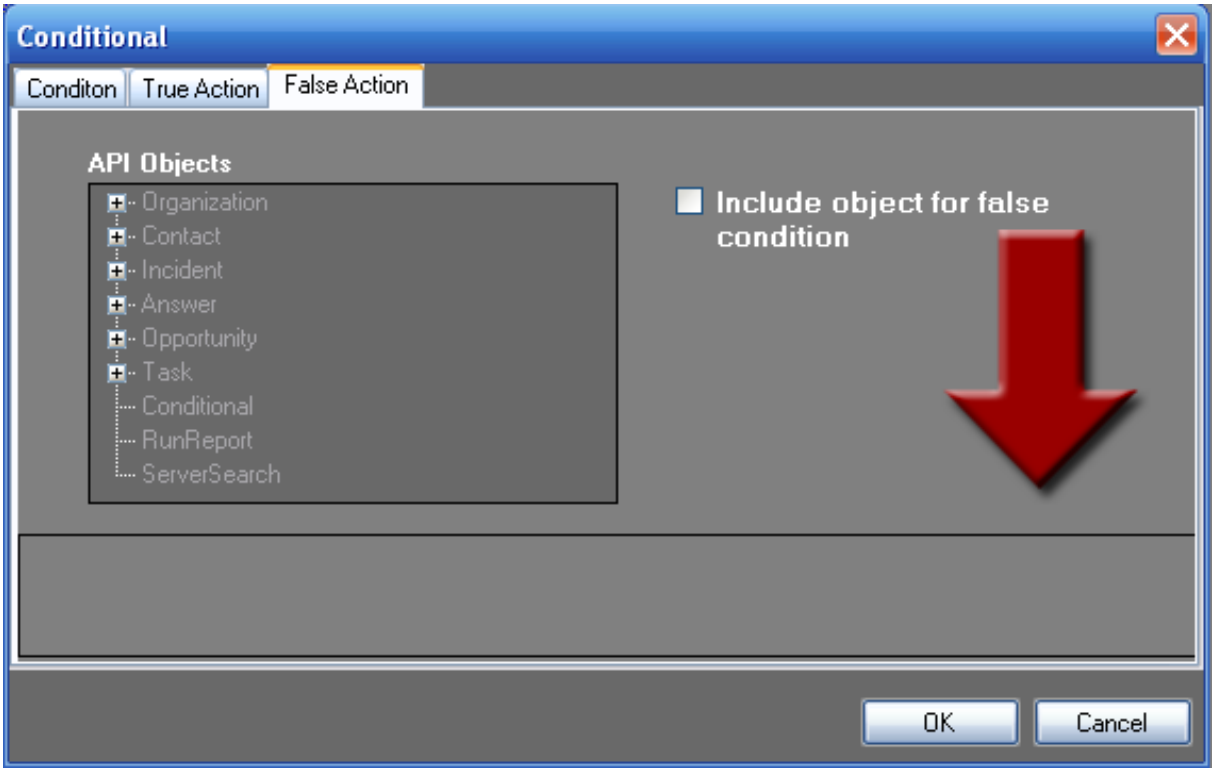
**Note:**
Although you can create a conditional step that checks if the token contains more than twenty results and then takes the true path, ScriptAssist currently has a limit of twenty screen pops. If a screen pop script requests more than twenty screen pops to occur at runtime, then an overflow condition is met and an error is traced to the logs. In this case, the rest of the script is abandoned. It is therefore important to create screen pop scripts that are focused and pop only a small number of records. You can create a pre-configured report in Oracle Service Cloud that pops if more than twenty records are returned.

4. On the **True Action** tab:




   a. Select the **Include object for true condition** check box.
   b. In the **API Objects** list, select the type of object to create if the conditional expression is true.

5. On the **False Action** tab:



a. Select the **Include object for false condition** check box.
b. In the **API Objects** list, select the type of object to create if the conditional expression is false.

6. Click **OK**.

# Order of script node execution

When you create or edit a script in the **ScriptAssist Script Creation** workspace, the script nodes will execute from the top left to the bottom right.

The current release of ScriptAssist does not support script branching or multithreading, so each script executes in a linear fashion.

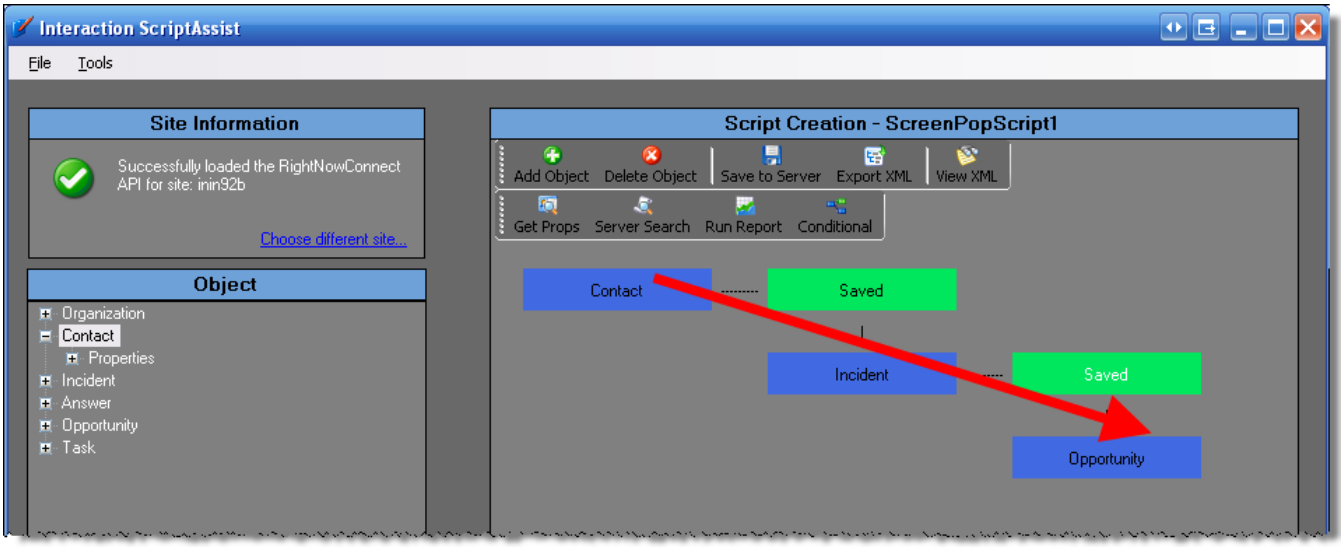
Execution flows from the first node, through all child nodes of that node; then to the next node, and through all child nodes; then to the next node, and so forth.

For example, suppose that you have one root node with a very large branch under it, and then a second root node (far down the script with the vertical dashed line). The second node will not be executed until the branch under the first node has finished executing. In this way, the script is entirely linear, even though it does not appear to be linear.

The following sections show three different examples of the order in which scripts execute.

## Linear execution with a single root node

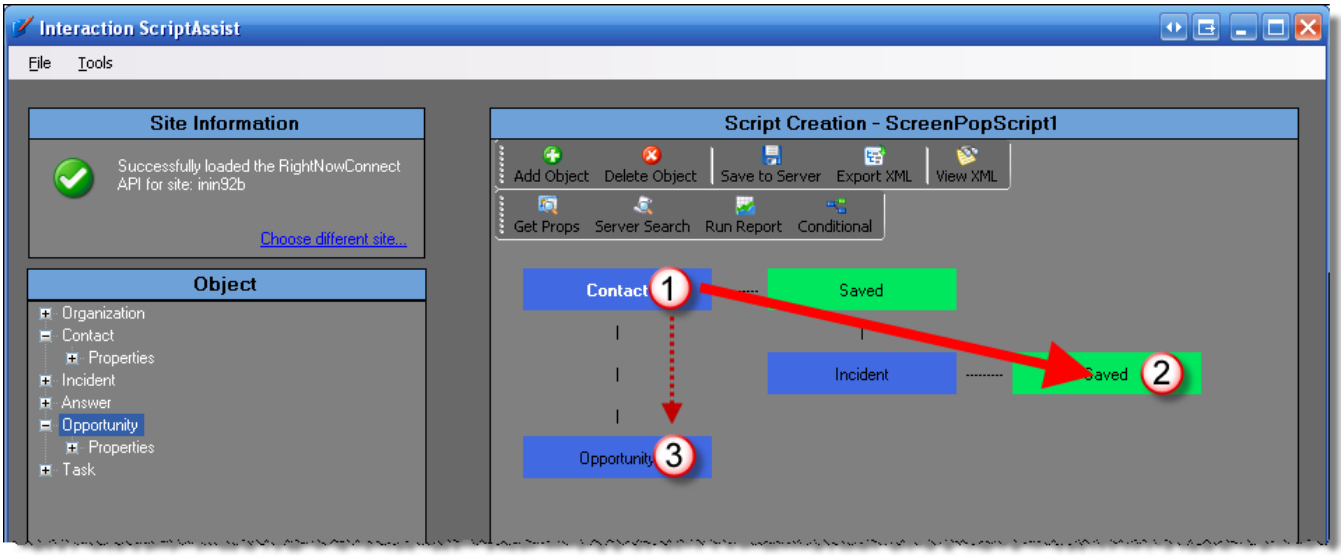
In a linear structure with a single root node, the script executes in linear fashion from the top left to the bottom right.




The script goes from **Contact** to **Saved**, **Incident**, **Saved**, and finally to **Opportunity**.

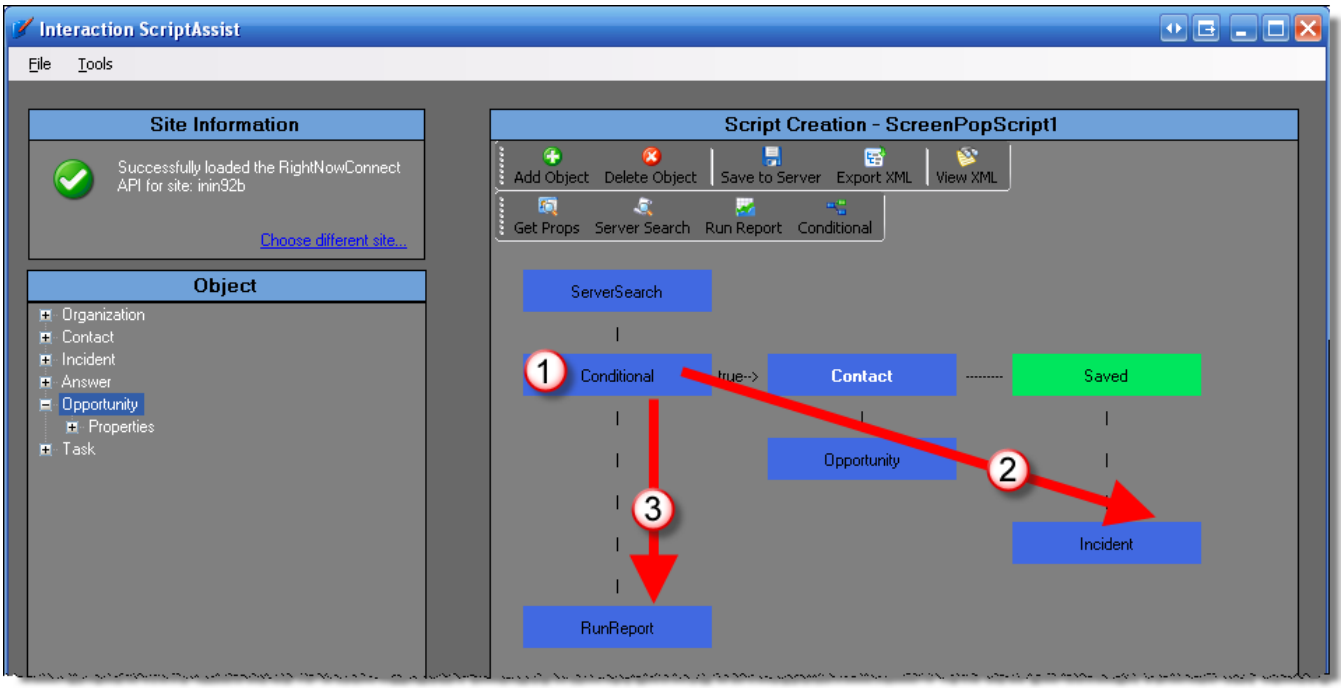
## Branched execution with an additional root node

In a branched structure with more than one root node, the script executes first from the initial root node (1) to the bottom right of the diagram (2). Only after it completes execution of this branch does it finally go to the other root node (3) and begin to execute that branch.




## Branched execution with a conditional node

In a branched structure with a conditional node, the script executes from top left to bottom right until it reaches the conditional node (1). At that point, the script follows one branch if the condition is true (2) and the other branch if it is false (3).




Note that when the script follows one branch of a conditional node, then the script will *not* execute the other branch of the conditional node.

# Configure custom actions

**Note:**
Genesys assumes that you know how to use Interaction Administrator and Interaction Attendant and that you have installed Interaction ScriptAssist. For more information about those programs, see the following:

- Interaction Administrator Help at https://help.genesys.com/cic/mergedProjects/wh_ia/desktop/interaction_administrator_help.htm
- Interaction Attendant Help at https://help.genesys.com/cic/mergedProjects/wh_iat/desktop/interaction_attendant_help.htm

Interaction ScriptAssist includes three ready-to-use script templates for screen pops:

- PopContactByANI
- PopIncidentByRefNo
- PopNewIncidentWithContactByANI

These templates are described in detail in Basics of scripts in ScriptAssist for Oracle Service Cloud.

To use these templates:

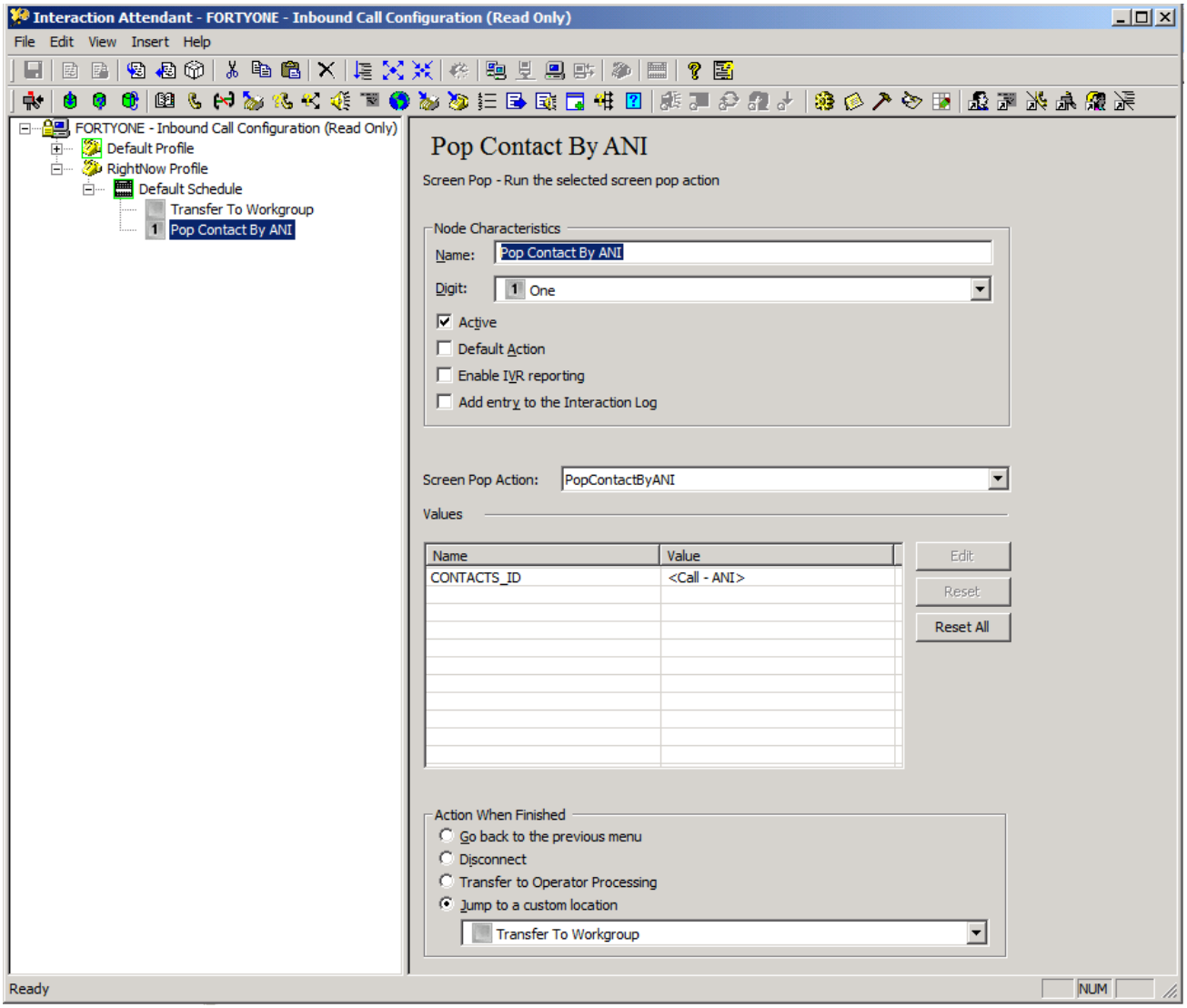
1. In Interaction ScriptAssist: Open the template and save it to the server.
2. In Interaction Attendant: Create a screen pop node and point it to an ACD workgroup.
3. In Interaction Administrator: Optionally assign a screen pop action directly to a workgroup.

## Enable Interaction Attendant to use the template script

The screen pop templates do not cause screen pops in the Oracle Service Cloud client until you inject them into a call flow. To do that, you can modify an Interaction Attendant profile to include the screen pop script.

To enable Interaction Attendant to use the script:

1. In Interaction Attendant on the IC server, locate the profile and schedule to which you want to apply the Oracle Service Cloud screen pop.
2. Create a screen pop node for the script that you want to set up:
   a. Click the schedule for the screen pop.
   b. On the **Insert** menu, point to **New Operation**, then click **Screen Pop**.
3. On the panel, complete the following steps.




4. Specify the node characteristics.
   a. Give it a name.
   b. Optionally give it a digit.
   c. Select the **Active** check box.
   d. Optionally select the **Default Action** check box.
5. Specify the appropriate screen pop action, such as **PopContactByANI**.
   a. Set the **Screen Pop Action**.
   b. Fill in the value for the attribute that is listed under the **Values** section.
6. Specify the action to take when the subroutine is finished.
7. The action usually transfers the interaction to a workgroup so that it can be routed to the Oracle Service Cloud client.

**Note:**
For more information about creating nodes, see Interaction Attendant documentation.

Some screen pop scripts must prompt the caller for information, and then set that information as an input value on the script. To accomplish this behavior:

- Use a **Caller Data Entry** node to prompt the caller for the required information.
- Store the data to a custom call attribute such as `IncidentIdAttr`.
- Using the data, set the name/value pair on the screen pop action in the **Screen Pop** node by specifying the call attribute name prepended with a dollar sign: for example, `$(IncidentIdAttr)`.

By doing this prompt work in Interaction Attendant, you can eliminate the need for custom handlers.

## Enable a workgroup to use the template script

By using Interaction Attendant to implement the screen pop script, you can include the script if your caller dials through a certain sequence of Interaction Attendant nodes.

Another implementation strategy takes advantage of the **ACD Alerting Action** field in the **ACD Actions** pane in the workgroup configuration. This action applies the screen pop script to any interaction that is added to the queue. To implement this strategy, modify the default values in the script and apply the script as an alerting action through Interaction Administrator.

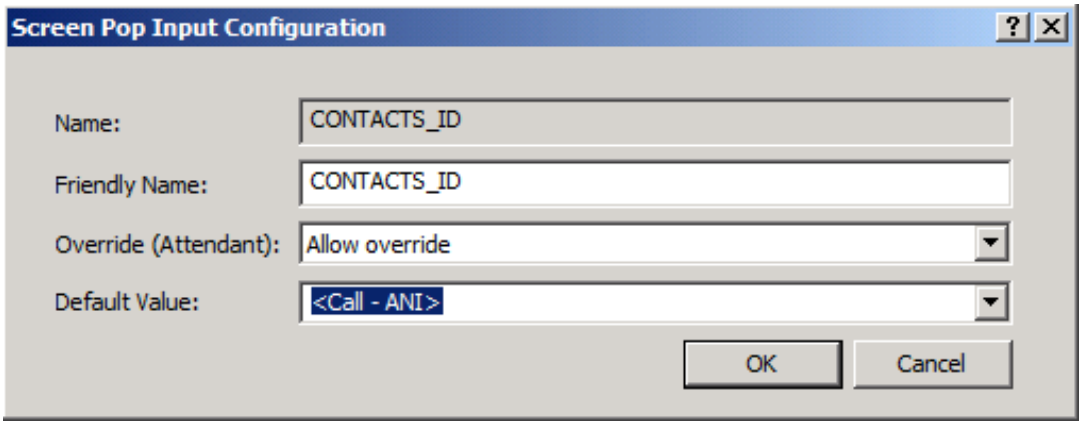
## Assign default values to script parameters

> **Note:**
> If you used Interaction Attendant to assign input parameter values, skip this section. Also, this step can be done before the Interaction Attendant step.

If you choose to apply the screen pop script directly to the workgroup by means of **Actions**, then update the script parameters to include default values.

To assign a default value using Interaction Administrator:

1. In Interaction Administrator, locate **System Configuration Actions**.
2. Open the screen pop script to which you want to assign default values.
3. Click the script input item and click **Edit**.
4. Enter or assign a default value to the script input item:



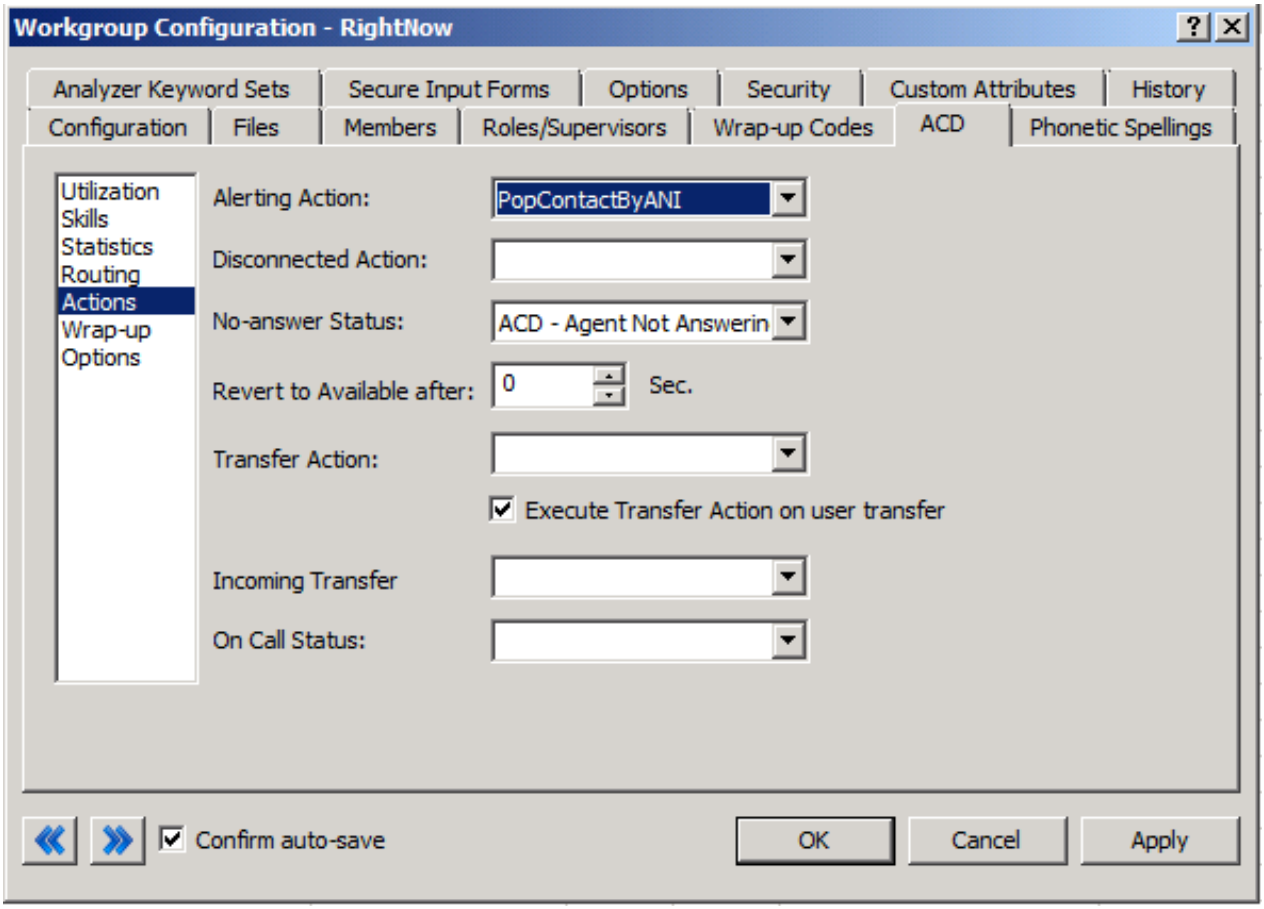

5. Click **OK**.

### Assign the screen pop script to a workgroup queue

After you have set the default values on the script, an alerting action can assign the script to a workgroup queue.

To assign the script to a workgroup queue:

1. In Interaction Administrator, locate the **Workgroups** node.
2. Open the **Workgroup Configuration** dialog box for the workgroup to which you want to apply the script.
3. Click the **ACD** tab.
4. In the list, click **Actions**.
5. In the **Alerting Action** list, select the screen pop script.




6. Click **OK**.

## Summary of steps and results

When you configured custom actions:

- You saved the PopContactByANI template script to the IC server from Interaction ScriptAssist.
- You updated the Interaction Attendant inbound call configuration. The configuration provides a way to apply the template screen pop script to the incoming calls using the **Screen Pop** operation in Interaction Attendant.
- You applied the screen pop script directly to a workgroup using the **ACD Actions** dialog box in Interaction Administrator.

With these tasks completed, screen pops work in the Oracle Service Cloud client. If a call comes in and the caller's ANI matches a contact record's phone number in the Oracle Service Cloud system, then that contact's record pops. If there is no contact to match the ANI, then no screen pop occurs.

# Customization points for Interaction Designer

Genesys has provided a number of customization points in the integration handler set. However, do not attempt customization unless you have advanced knowledge of CIC handlers and intimate "under the hood" experience with the IC Integration with Oracle Service Cloud.

## Handle the incident screen pop script for routed interactions

The Integration Service allows you to route Oracle Service Cloud incidents through the CIC platform using the ACD routing engine. By default, the integration looks for the incident screen pop script to apply to routed interactions so that the routed interaction is automatically screen popped on the agent desktop. The handlers locate this default script in one of two ways:

- They first look for an Oracle Service Cloud screen pop action in DS with name: `<Queue Name>_QueueEmailScript`. You can create queue-specific, incident screen pop scripts based on the CIC queue to which the Oracle Service Cloud incident is routed.
- If the handlers do not find a queue-specific incident script, then they look for an Oracle Service Cloud screen pop action called `DefaultEmailScript` and apply that script. The integration installs this default screen pop script automatically.

# Change log

The following table lists the changes to the *Interaction ScriptAssist for Oracle Service Cloud Technical Reference* since its initial release.

| Date | Changes |
|---|---|
| 12-July-2012 | Updated copyrights and made minor tweaks. |
| 24-January-2013 | • Removed Using Advanced Mode section.<br>• Updated Preferences screenshot. |
| 09-September-2014 | • Updated documentation to reflect changes required in the transition to CIC 2015 R1, such as updates to product version numbers, system requirements, installation procedures, references to Interactive Intelligence Product Information site URLs, and copyright and trademark information.<br>• Updated product name from RightNow to Oracle Service Cloud.<br>• Modified three screenshots.<br>• Added and updated Using Advanced Mode section. |
| 29-September-2014 | • Changed the title to Interaction ScriptAssist for Oracle Service Cloud.<br>• Updated all references to RightNow that needed to be updated. |
| 03-December-2014 | • Added most of Chapter 2, Using the Templates.<br>• Added Chapter 4, took it from the Install and Config guide of the Interactive Intelligence Integration with Oracle Service Cloud.<br>• Updated the RightNow references to Oracle Service Cloud. |
| 03-February-2015 | Added Support of the Oracle Service Cloud UI section. |
| 10-April-2015 | • Added Search results section, which explains what happens when search results are greater than 20 items.<br>• Added a note in the Use a Run Report node section about adding filters.<br>• Updated for 2015 R3. |
| 30-April-2015 | Removed section Support of the Oracle Service Cloud UI. |
| 04-August-2015 | Updated for 2015 R4. |
| 09-October-2015 | Updated for 2016 R1. |
| 20-October-2015 | Updated info in Search results section, which explains what happens when search results are greater than 20 items. |
| 21-December-2015 | Updated for 2016 R2. |
| 11-May-2016 | Updated Title page and Copyright and Trademarks page. |
| 02-May-2018 | Rebranded from Interactive Intelligence to Genesys. |
| 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...". |