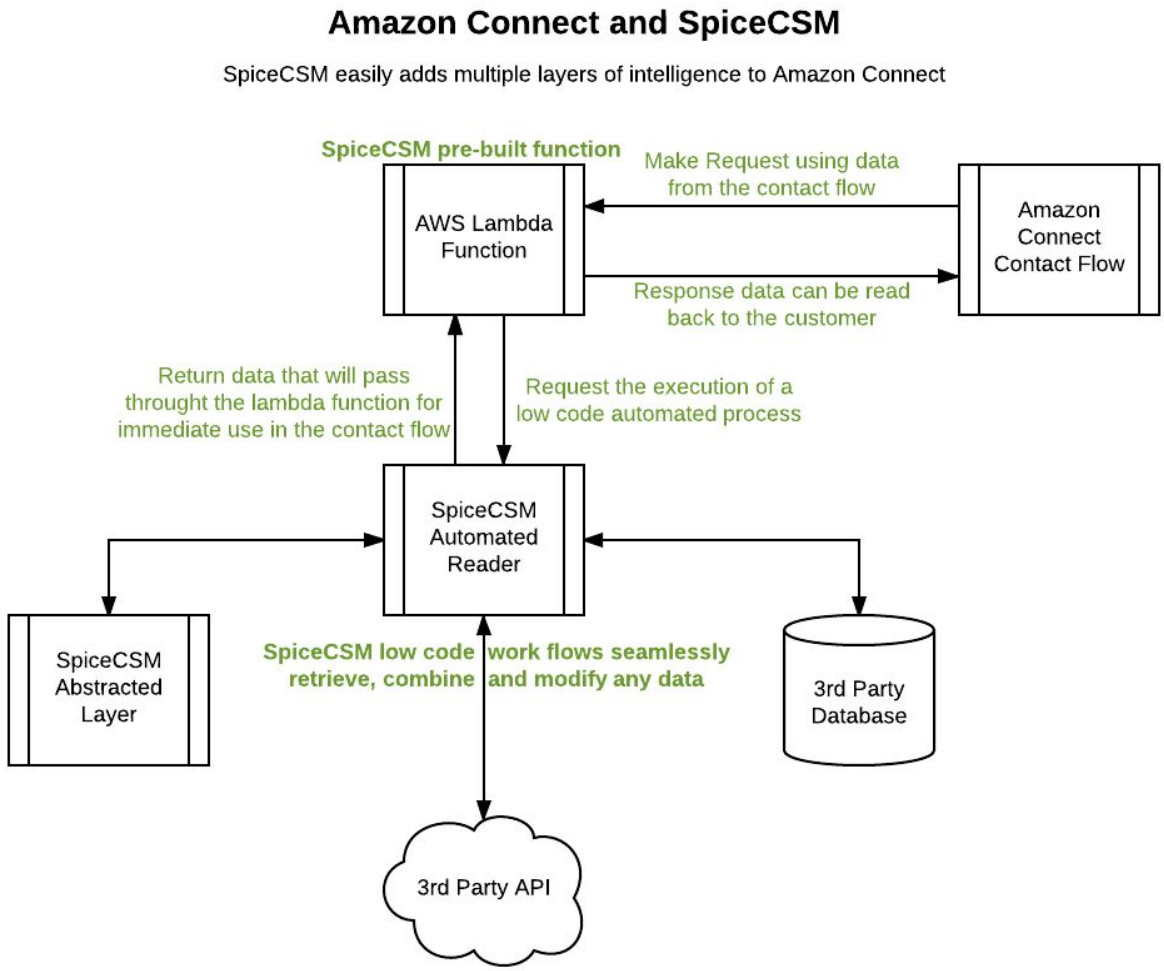
Amazon Connect - SpiceCSM Automated Reader Integration

User Guide

Overview

Amazon Connect and SpiceCSM together allow for the rapid development of intelligent IVR systems. The general flow in which Amazon Connect interacts with SpiceCSM is depicted in the following diagram.



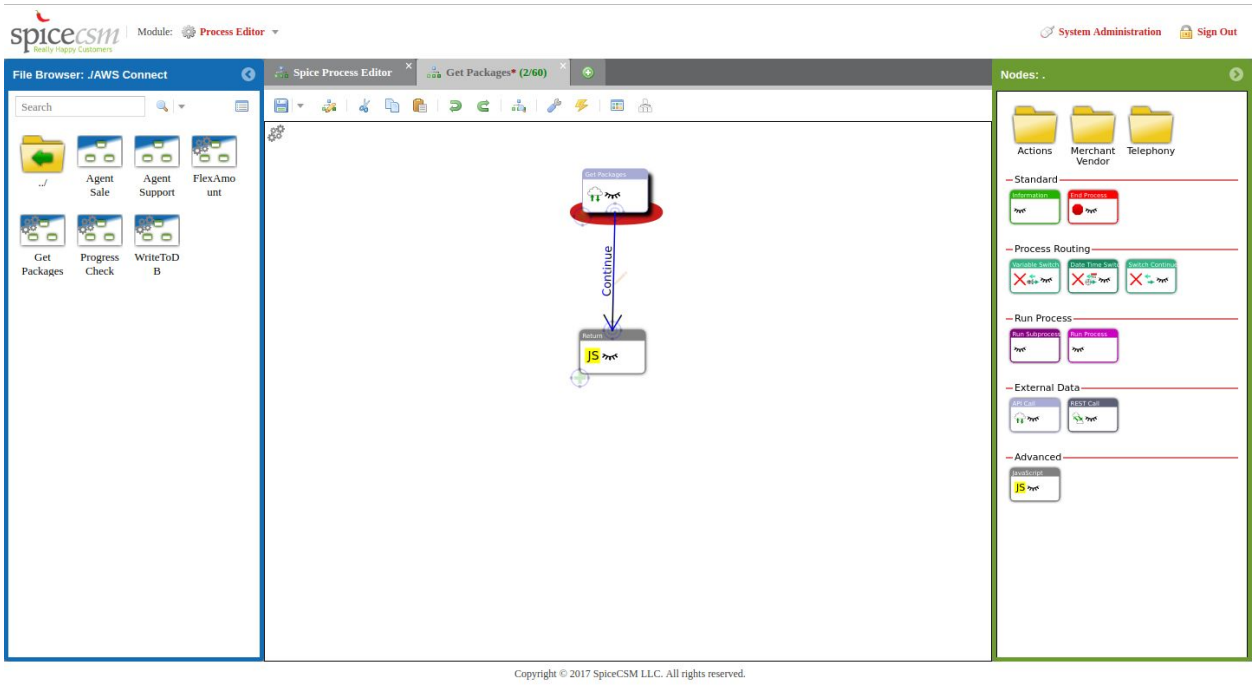
Prerequisites

Access to an instance of Amazon Connect and a SpiceCSM subscription which has access to Automated Processes is required to utilize the integration between Amazon Connect and the SpiceCSM Automated Reader. Some knowledge of programming will be helpful.

Setup AWS Lambda Function via a QuickStart

Automated Process Overview

Screenshot 1

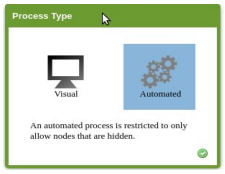


The above screenshot shows an Automated Process. In the upper right corner of the canvas (under the save button on the toolbar) there is an icon that shows the user that the current process is an Automated Process. That same icon is displayed on the file in the File Browser for the same reason.

Automated Processes run with no human interaction so all the nodes for this process must be hidden. When a process like this is created all the nodes in the Node Browser are automatically hidden and nodes that cannot be hidden are removed.

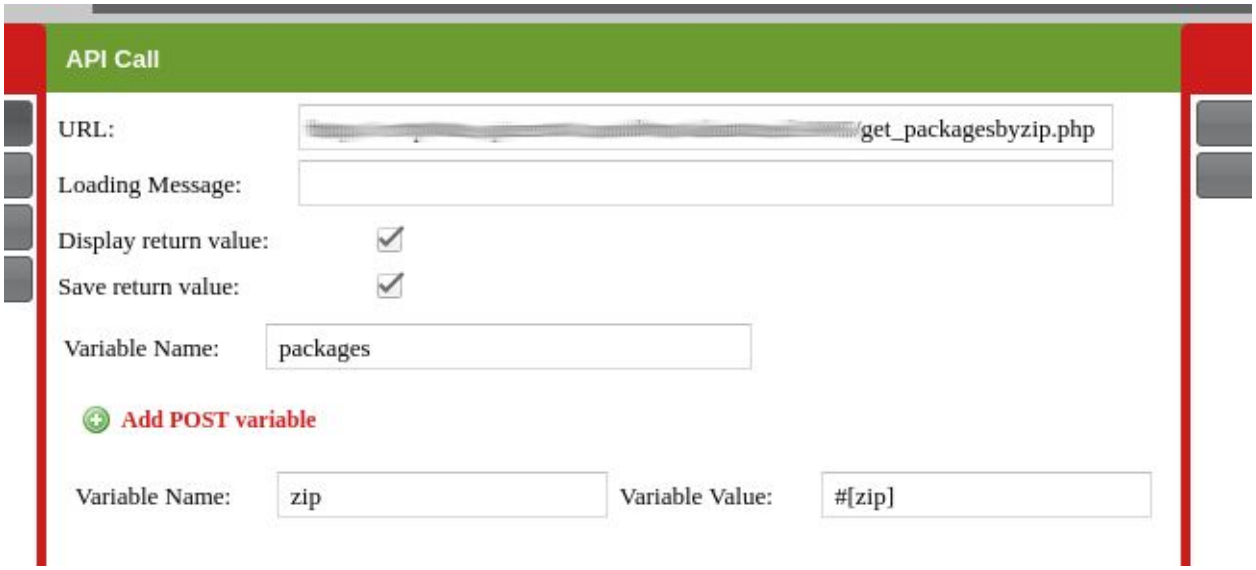
The Automated Process must be created before an Amazon Connect Contact Flow can ‘tell’ the Automated Reader execute it. The next section will briefly describe the process in the above screenshot.

Creating an Automated Process



When the Process Editor loads, a window will open where the process type can be selected. If this does not occur then the current SpiceCSM subscription does not allow for Automated Processes. Alternatively, the SpiceCSM subscription could only be for Automated Processes. Check the upper left corner of the canvas for the automated icon.

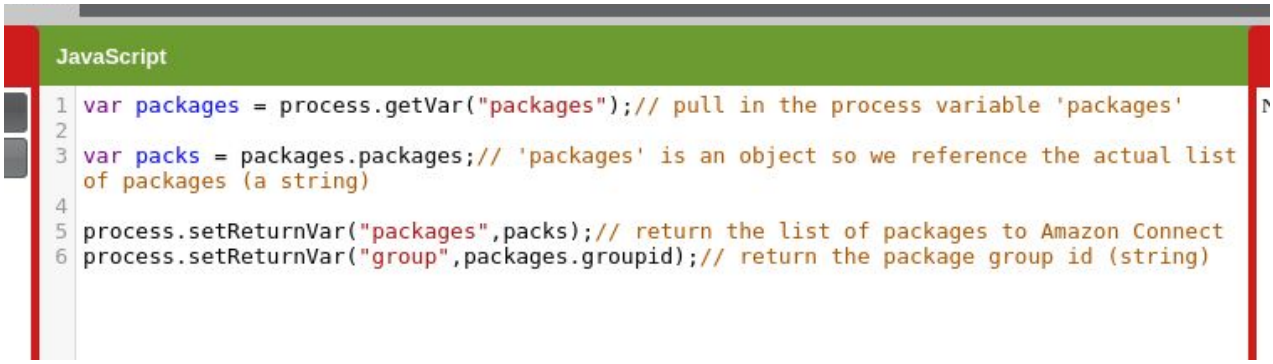
The majority of Automated Processes will most likely need to reach out to a third party system in order to retrieve some data that will ultimately be returned to the Connect Contact Flow. This can be accomplished with the API Call node and the REST Call node. The process in Screenshot 1 has an API Call node as the root. The root node is where the process starts. So in the example, the first thing that occurs is a call to import more data into the process. In this case the call is to a third party system to retrieve a list of packages based on the callers zip code. To edit the node, double click on it. Below is a screenshot of the node's settings.



The URL, POST Variable, and a process variable to save the returned data are specified in the settings of the node. The 'Display return value' checkbox will show part of the returned data in the log for the Automated Process (explained later). The #[ ] syntax is used for interpolation of process variables. So the value of the process variable zip will be inserted as the value for the POST variable zip.

The data saved by the API Call node may need to be formatted and set to return when the process is finished executing. The JavaScript node can format the data and set the formatted

data to be returned to the Contact Flow. Below is a screenshot of the JavaScript node's settings.

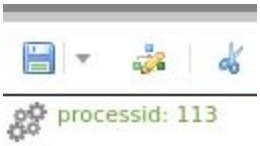


Each JavaScript node is self contained and does not know about any other JavaScript node. The scripts are executed when encountered in the process reader. The node has a predefined variable 'process'. It is used to access process variables and set variables to be returned from the Automated Reader.

The Amazon Connect Contact Flows only allows for the return of key value pairs where the value is a string. The above script is importing the process variable 'packages' which is an object. So the actual list of packages, which is a string, can be reassigned to another variable (line 3 above) or can be set to return directly (line 6 above for groupid). This node executes any valid JavaScript, so strings can be dynamically constructed and returned to the Contact Flow to be branched on or read back to the caller.

If there is no data processing required, the Return Data node can be used to easily return the appropriate data. The 'Variable Value' field support interpolation so the value of the variable will be passed back to the Contact Flow.

Locating the Process ID



The process id is required for the execution of the process from the Contact Flow. The process id can easily be located after saving the process. As seen in the screenshot, the process id is located in the upper left of the canvas next to the automated icon. The process id will have to be entered as a parameter in the 'Invoke AWS Lambda function' block in the Connect Contact Flow.

Run an Automated Process from an Amazon Connect Contact Flow

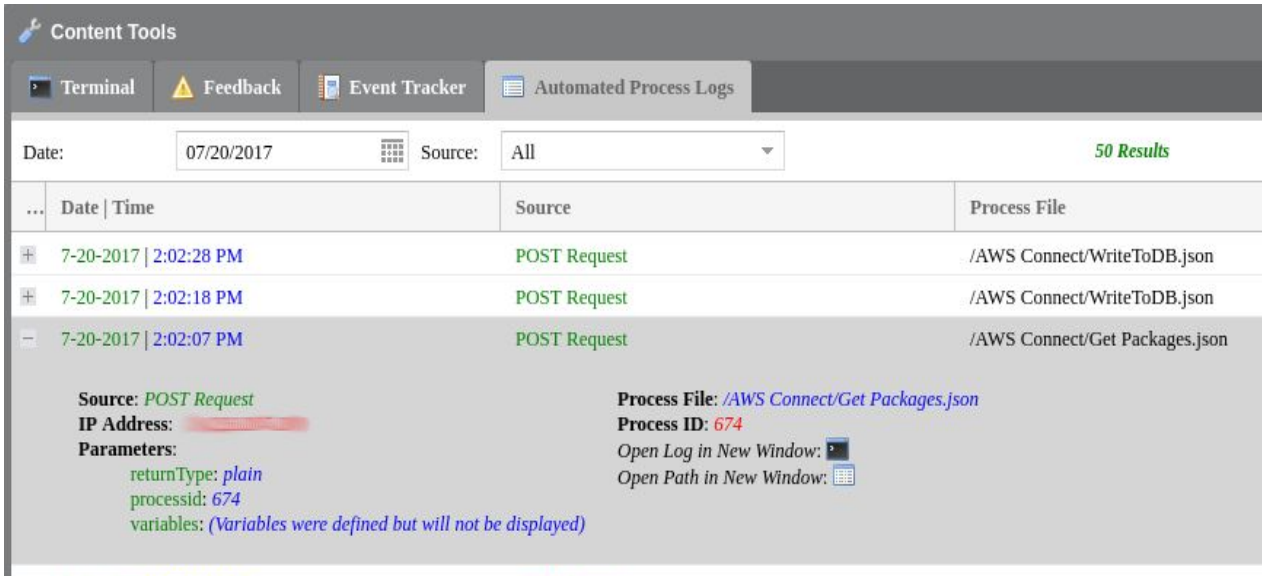
To run an automated process from a Contact Flow simply use the 'Invoke AWS Lambda function' block. Specify SpiceCSM's pre-built lambda function for the 'Function ARN', add input parameters for the 'processid' and any other parameters that need to be passed to the automated process.

In this screenshot, the processid and the customer's phone number are being sent to the Automated Process. The customer's phone number has a key of 'ani' so in the process the value can be referenced by using the syntax *#[ani]* or imported to a JavaScript node by using *process.getVar('ani')*.

When the contact flow is executed, the lambda function will be called and any variables returned from the process (via the JavaScript node's setReturnVar function) will be saved to the contact flow's External attribute. For the packages example, to read the list of packages back to the caller enter *$.External.packages* into any block that can execute the text to speech functionality (Play prompt, Get customer input, …).

Examine Automated Process Execution

The SpiceCSM pre-built lambda function has built in logging for Amazon CloudWatch. From there the parameters passed to and from the Automated Process can be viewed. The Automated Process Logs can be used to see exactly what happened in the process. There is a button on the toolbar that looks like a wrench. This button opens the Content Tools window. The logs can be viewed under the *Automated Process Logs* tab.



The entry for the executed process has useful information to help diagnosis any issue that may occur. The most helpful feature is having the ability to open the process log in a new window. Click on the icon to open the window.

Source: *POST Request*
IP Address: [redacted]
Parameters:
returnType: *plain*
processid: *674*
variables: *(Variables were defined but will not be displayed)*

Process File: */AWS Connect/Get Packages.json*
Process ID: *674*
Date | Time: *7-20-2017 | 2:02:07 PM*

```
14:02:07 :: Tree initialized for file "Get Packages"
14:02:07 :: Current tree = Get Packages
14:02:07 :: Current node = Get Packages
14:02:07 :: Current process id = 674
14:02:07 :: Passed in variables initialized
14:02:07 :: History initialized.
14:02:07 :: TreeReader Created Successfully
14:02:07 :: Processing root node for tree "Get Packages" with node label "Get Packages" and id "2Up3w10m82Fr2c4t7c10w5t4n6t9"
14:02:07 :: Processing node "Get Packages" with id "2Up3w10m82Fr2c4t7c10w5t4n6t9"
14:02:07 :: Loading remote data ... | URL: '[redacted]/get_packagesbyzip.php
14:02:07 :: Displaying part of the returned data: {packages: Press 1 for Basic DSL, Press 2 for Extreme DSL,Press 3 for Ultra D
14:02:07 :: Attempting to save returned data
14:02:07 :: Saving variables as an array
14:02:07 :: Setting Variable: packages['packages'] to Press 1 for Basic DSL, Press 2 for Extreme DSL,Press 3 for Ultra DSL
Result: packages['packages'] = Press 1 for Basic DSL, Press 2 for Extreme DSL,Press 3 for Ultra DSL
14:02:07 :: Setting Variable: packages['groupid'] to 2
Result: packages['groupid'] = 2
14:02:07 :: Checking for completion of async variables -> Status, Issue, Subissue
14:02:07 :: Async variables set
14:02:07 :: Processing node "Get Packages" with id "2Up3w10m82Fr2c4t7c10w5t4n6t9"
14:02:07 :: Hidden node detected, automatically advancing on path "Continue" to node "Return"
14:02:07 :: Hidden history choice added for node "Get Packages" with id "2Up3w10m82Fr2c4t7c10w5t4n6t9". Path taken "Continue" to node "Return"
14:02:07 :: Executing node callback
14:02:07 :: Checking for completion of async variables -> Status, Issue, Subissue
14:02:07 :: Async variables set
14:02:07 :: Processing node "Return" with id "5G9Ow53G8Hv0k40A4J6D10Iu0m2"
14:02:07 :: The JavaScript Node with label Return and id 5G9Ow53G8Hv0k40A4J6D10Iu0m2 had no output
14:02:07 :: Hidden end node -> Ending Process
14:02:07 :: Hidden history choice added for node "Return" with id "5G9Ow53G8Hv0k40A4J6D10Iu0m2" Finished Tree
14:02:07 :: Checking for completion of async variables -> Status, Issue, Subissue
14:02:07 :: Async variables set
14:02:07 :: Executing Finish Callback -> Ending Process
```