Contact CRM Integration Developer Guide

Introduction

Horizon Contact is a web-based customer contact solution designed to improve the quality of conversations your business has with its customers.

Horizon Contact includes a CRM to track all these conversations to give a full history for each customer, however, many businesses have already invested in a CRM package and so Horizon Contact provides some tools to enable you to integrate this and record those customer interactions in the CRM you are already using.

Pre-Requisites

To use data from your CRM with Horizon Contact you need to make sure that your CRM is accessible over the internet i.e., via the https protocol. It also needs to be able to receive and process REST queries and deliver information in JSON format. Usually, your CRM provider will have a section on their website detailing the structure of the queries it will accept and the information it can exchange, for example, here is a link to the Zendesk developers pages: API Reference Home | Zendesk Developer Docs

Structure of this Document

In this document we will show you:

- An overview of the process to interact with a CRM
- A detailed step-by-step example using Zendesk
- A complete list of the nodes and their syntax to enable you to build your own integrations

Process Overview

What can I query and how can I use this information?

This will depend to some degree on the CRM or application you are trying to integrate with, so again we urge you to consult the relevant developer documentation.

The way Horizon Contact interacts with these applications is through building interaction flows with some specific nodes which can send queries and accept information back from the application. It can then use this information to perform actions directly in the application.

Here are some examples of what you can do using the tools within Horizon Contact.

- Store customer information, e.g., a phone number to construct REST queries within your interaction flows.
- Pass login credentials to that application so that agent the does not need to login in or authenticate each time the application is queried.
- Send a REST query to a website to discover whether the customer exists already within the CRM.
- On connecting with the customer, either open a blank record, if the customer does not exist or else open the customer's record within the third-party CRM. This is often known as screen popping.
- Depending on the application you may be able to show any open cases or records relating to that customer.
- Pass classification codes back to the application on completion of the interaction.

In these examples, management within the channels, e.g., call transfer would remain in the Horizon Contact application.

High-Level Overview of Building a Call Flow to Integrate with Zendesk

The following call flow can be used to open a customer's record within Zendesk when an agent is connected to the customer.

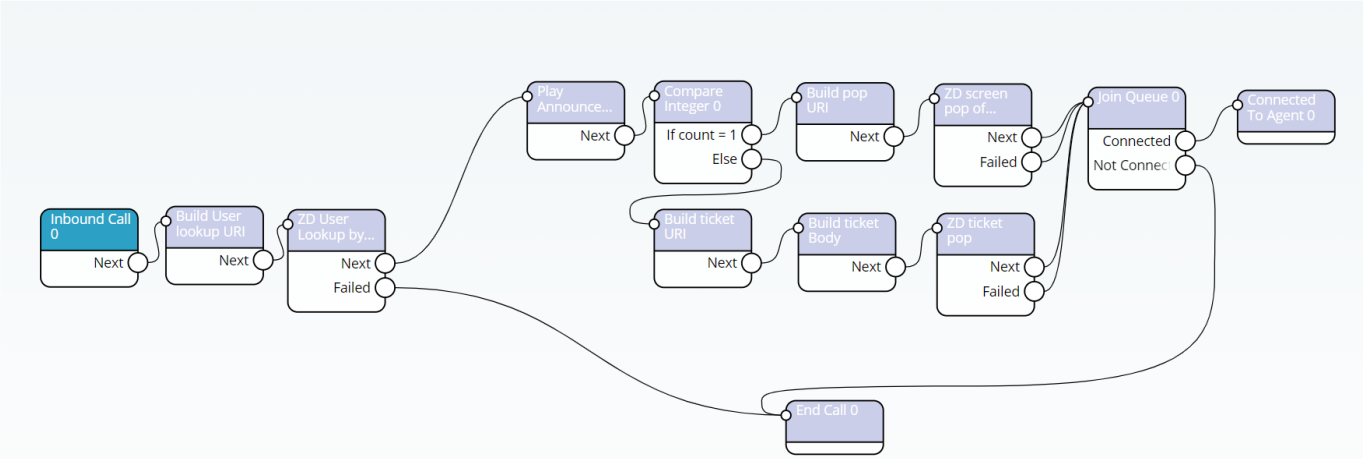

Figure 1. Example of Call flow built within Horizon Contact

This is how it works:

- The Inbound Call node is used to store the customer's CLI when they call the service number.
- This is passed to the Build User Lookup URL node which is what we have called the String Builder. Here a combination of the CLI collected in the first node and a custom address is used to build that string that will be used in the query. For the exact syntax required, you should refer to your third-party application's documentation
- Now that we've constructed the query, we can send the REST query to Zendesk and store the result as a JSON object

In this example we then play an announcement to the caller

- We then use the Compare Integer node to check whether there was a match between the customer's CLI and a record in Zendesk.
- If there is a match, we generate another string to pass to Zendesk using the String Builder node, here called Build Pop URL
- We then prepare to pass that string to Zendesk when the agent answers. We do this with a Prepare REST Query node that holds on to that query whilst
- The call is placed on a queue and then
- Answered by the agent, whereupon the customer's record is shown on Zendesk on the agent's desktop

In the case that there is no customer record a blank record is opened and so when the call is connected, a new contact screen is opened on Zendesk.

Step by Step Example – Screen Pop a Contact in Zendesk.

In this example, a query is made to Zendesk to check if the Phone Number of an inbound call is associated with a contact in Zendesk. If this is the case, then a request is made to Zendesk to open that contact as the call is presented to an agent.

Firstly, find the API documentation for Zendesk. The following two documents are useful.

https://developer.zendesk.com/apps/docs/developer-guide/getting_started

https://developer.zendesk.com/rest_api/docs/talk-partner-edition-api/introduction

We will be making use of the following API's:

Search API to find the contact from the phone number (Search | Zendesk Developer Docs)

/api/v2/search.json?query={search_string}

Open a User's Profile in an Agent's Browser (Talk Partner Edition API | Zendesk Developer Docs)

/api/v2/channels/voice/agents/{agent_id}/users/{user_id}/display

The sample call flow looks like this, and can be downloaded as a reference.

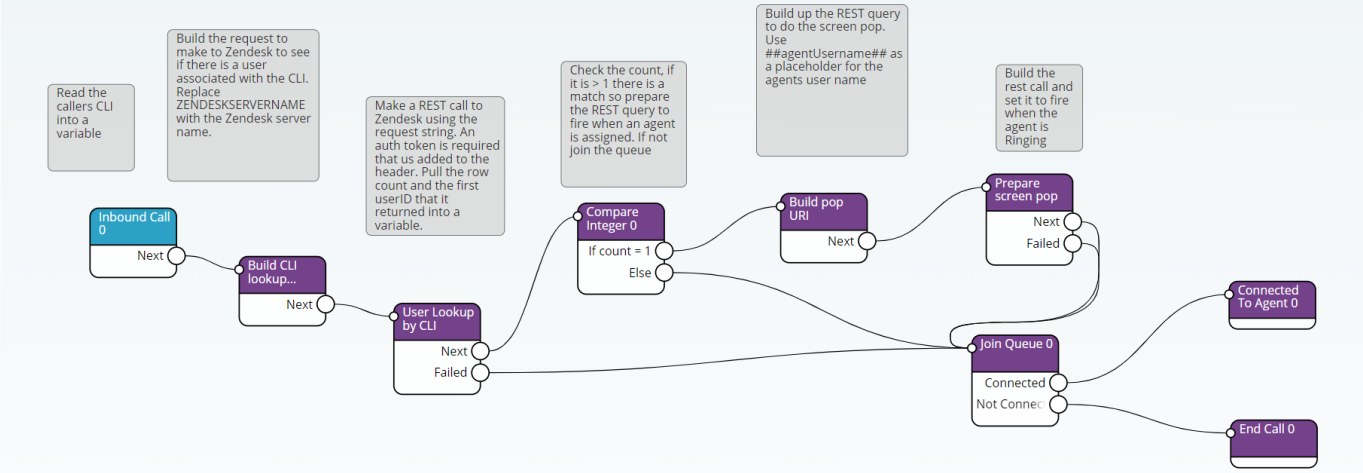

Figure 2. Example of Call flow with notes.

Retrieve the Contact’s Number

This is done in the Inbound call node. In this node the Caller’s number is assigned to a variable for use in later nodes.

The number is assigned to variable called userCLI.

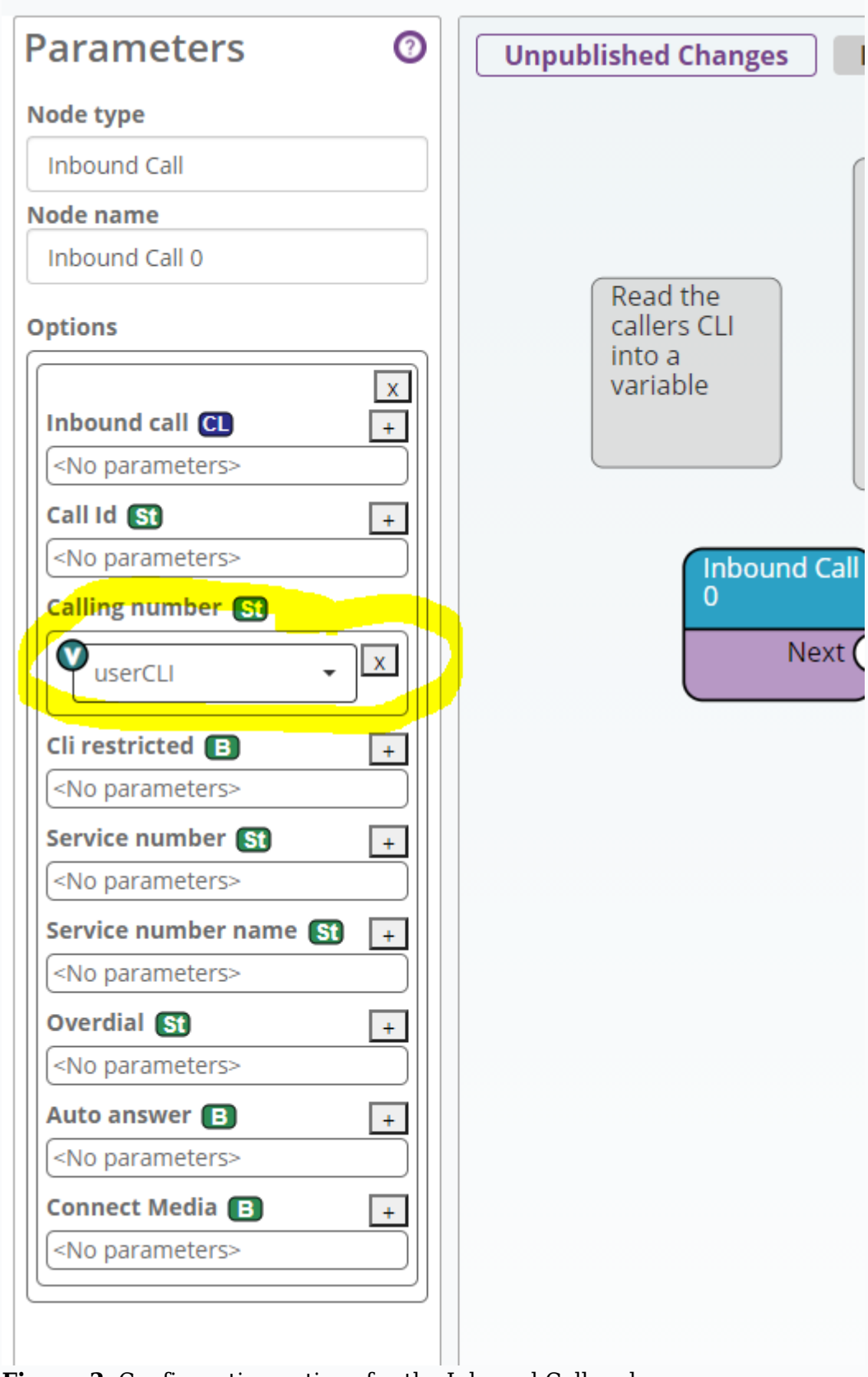

Figure 3. Configuration options for the Inbound Call node

Use a String Builder Node to Construct the API Request Using the CLI

The request to Zendesk must look like this

https://ZENDESKSERVERNAME.zendesk.com/api/v2/search.json?query=type:user phone:<CLI>

Where <CLI> is the incoming phone number.

This can be constructed by using the string builder node to attach the userCLI variable from step one to a string containing the query. As the URL above has spaces in it, it must be URL encoded, so this option is selected.

The full request containing the CLI is assigned to a variable called SearchURI

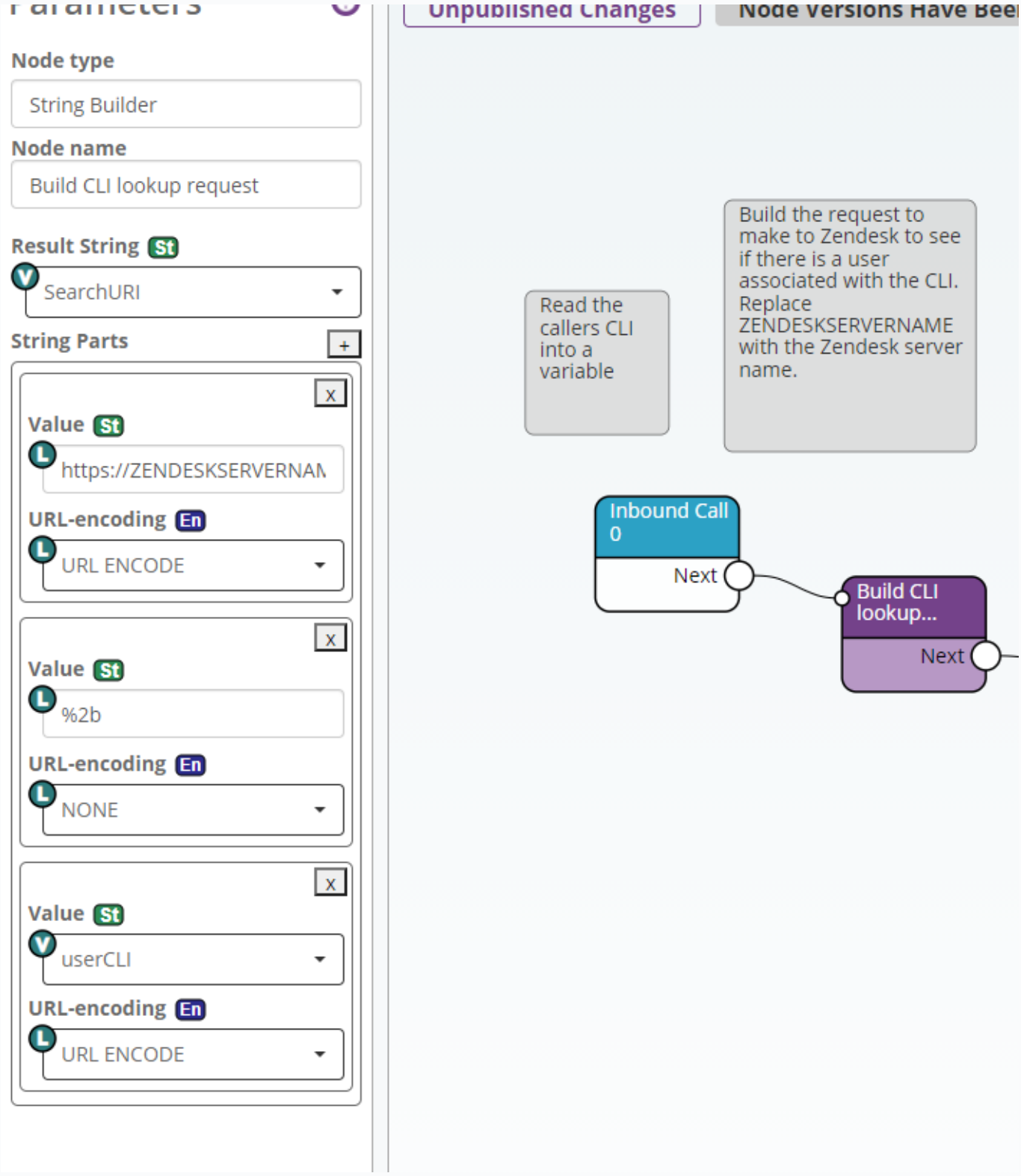

Figure 4. Configuration options String Builder node

Make the Call to Zendesk Using a REST Query Node

The searchURI variable from step 2 is used for the request. The request will return a JSON object, and the values of interest (count and id) must be extracted from this object.

Note on Extracting values from a JSON Object.

To extract a value from a JSON file, we need to provide the path to the record of interest. As a JSON document may have nested values using Curly and square brackets, there is a syntax to navigate to a particular row. Curly brackets denote a child record, so we must add the name of the child record. Square brackets denote a set of rows, so we must refer to the row in the set we are interested in.

Let us start with an example. Given this JSON;


```
{
  “result”: {
    “success”: true,
    “details”: {
      “code”: 100,
      “message”: “Success”
    }
  },
  “data”: [
    { “name”: “Alice”, “age”: 27 },
    { “name”: “Bob”, “age”: 62 }
  ]
}
```


Given the above data, the following paths would result in the following values being written to the provided variable:

Path	Type	Value
result.success	Boolean	True
Result.details.message	String	Success
Data.0.name	String	Alice
Date.1.age	Integer	62

The request will return a result that looks like this:

```
{
    "count": 1,
    "facets": null,
    "next_page": null,
    "previous_page": null,
    "results": [
      {
        "created_at": "2018-04-06T03:17:05Z",
        "default": false,
        "deleted": false,
        "description": "",
        "id": 1835972,
        "name": "Ragtail",
        "result_type": "group",
        "updated_at": "2018-04-06T03:17:05Z",
        "url": "https://example.zendesk.com/api/v2/groups/1835972.json"
      }
   }
```


The count will tell us how many records have been returned. The id is the contact's id, and this is what we will use for the screen pop. If the CLI record matches more than one record, then we will always take the first one returned.

The count is written to a variable called intcount. As the count is in the main document, we can reference it just as count.

As the id could be in the results section, and there could be multiple results sections, we refer the first result. So, the path is results.0.id

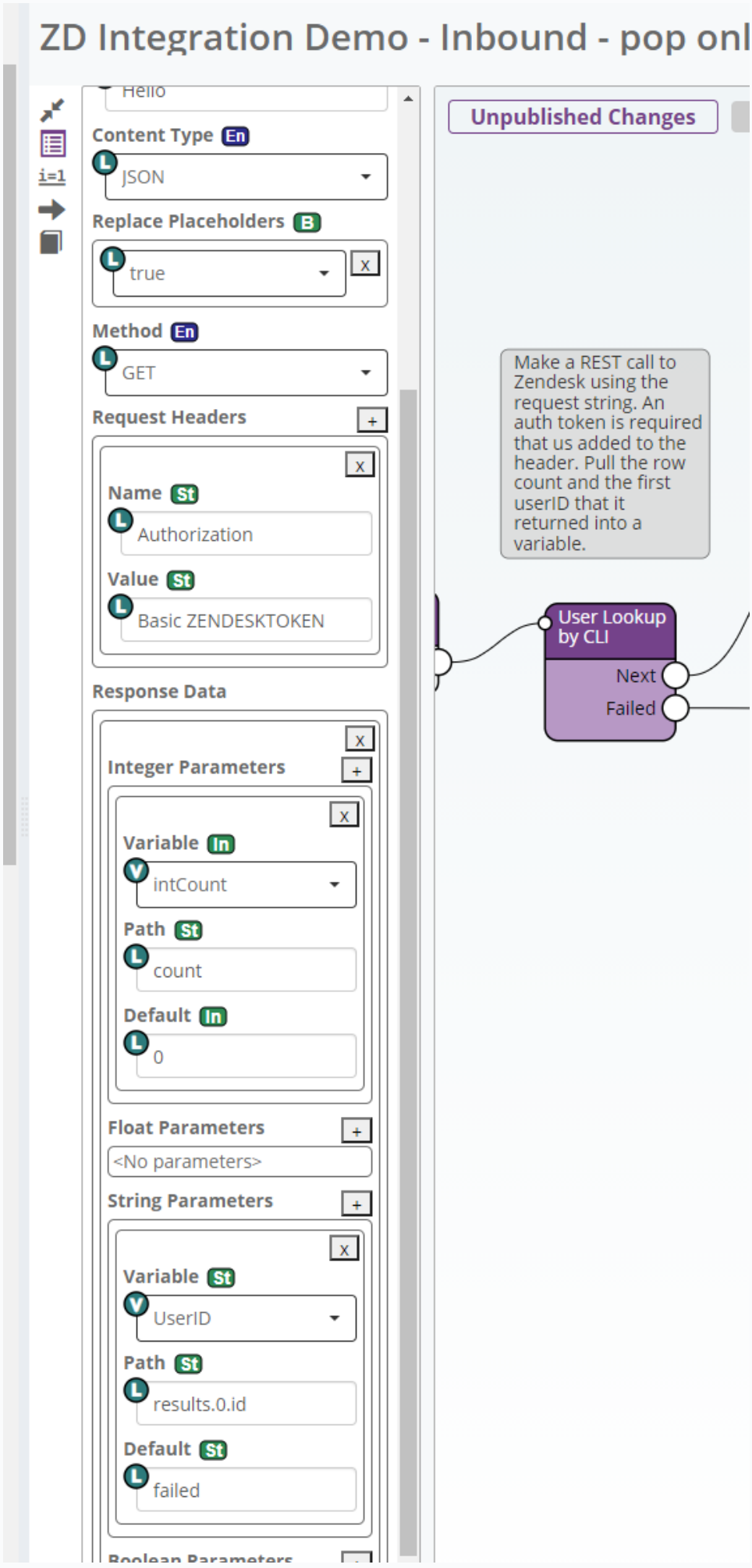
ZD Integration Demo - Inbound - pop onl
Hello
Content Type En
JSON
Replace Placeholders B
true
Method En
GET
Request Headers
Name St
Authorization
Value St
Basic ZENDESKTOKEN
Response Data
Integer Parameters
Variable In
intCount
Path St
count
Default In
0
Float Parameters
<No parameters>
String Parameters
Variable St
UserID
Path St
results.0.id
Default St
failed
Unpublished Changes
Make a REST call to Zendesk using the request string. An auth token is required that us added to the header. Pull the row count and the first userID that it returned into a variable.
User Lookup by CLI
Next
Failed

Figure 5. REST Query node configuration

To make this request to Zendesk, the query requires a valid token. First, generate a token in Zendesk as per these instructions:

[Generating a new API token – Zendesk help]

Then add it to the request header section of the request as per the above screen capture.

Check to see if the contact exists in Zendesk

Use a compare integer node to check that the count value returned was 1. If the contact does not exist, we miss the next steps as we do not gave a contact record to open on call arrival.

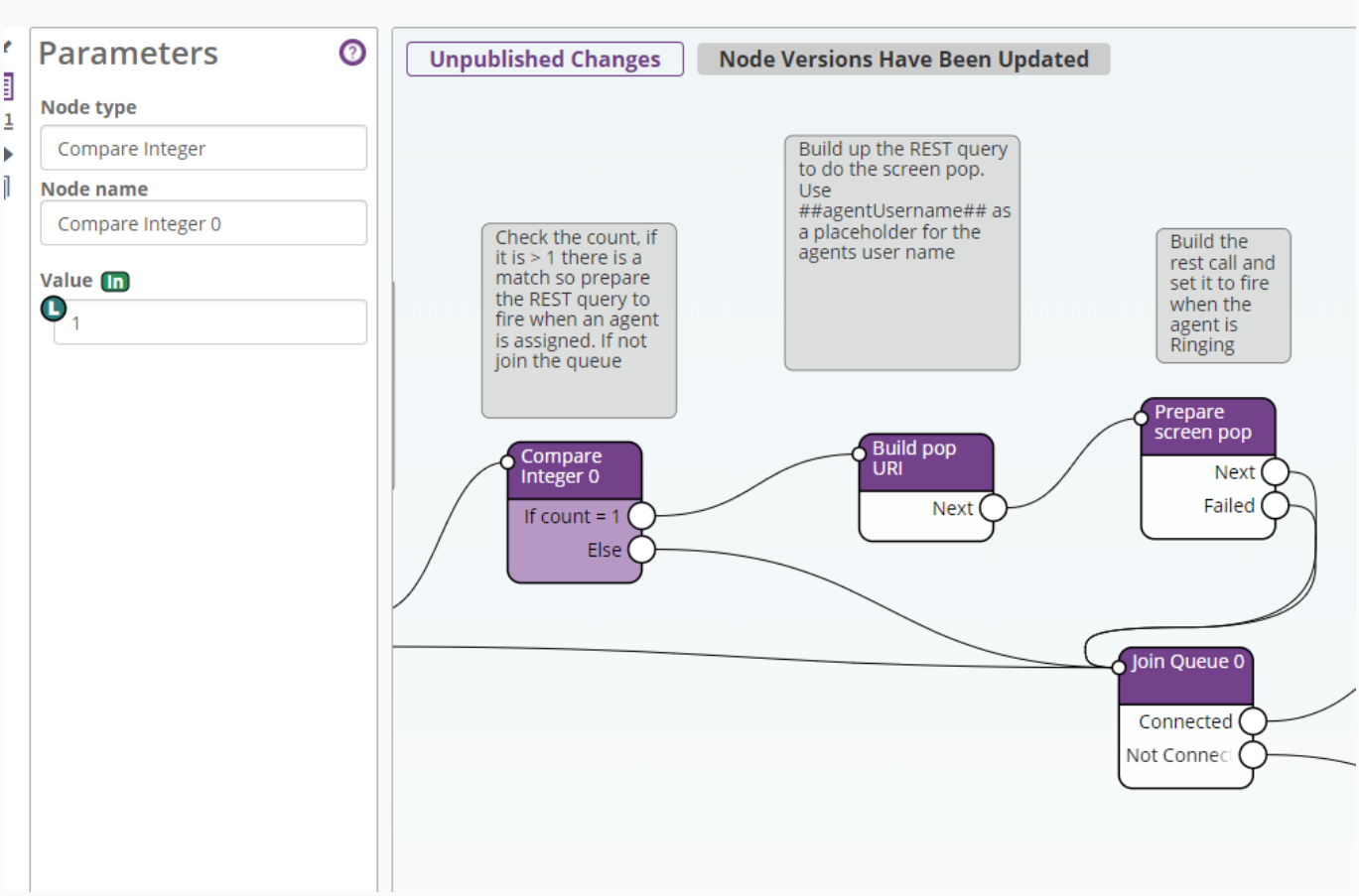

Figure 6. REST Query node configuration

Build the URI to do the screen pop

The final request must look like this:

https://ZENDESKSERVERNAME.zendesk.com/api/v2/channels/voice/agents/<Agent ID>/users/<Contact ID>/display.json

Where <Contact ID> is the contact id (we are using the UserID variable retrieved in an earlier step, and <Agent ID> is the Zendesk Agent ID.

At this stage we know who the contact is in Zendesk, but we don't yet know which agent will be assigned to the call, so we build the URI using placeholder for the agent. In this case we are using ##agentUsername## as a placeholder, it is also possible to use the Agents External ID, if this is the

field that holds the Zendesk agent ID.

The complete URI is written into a variable called ContactURI

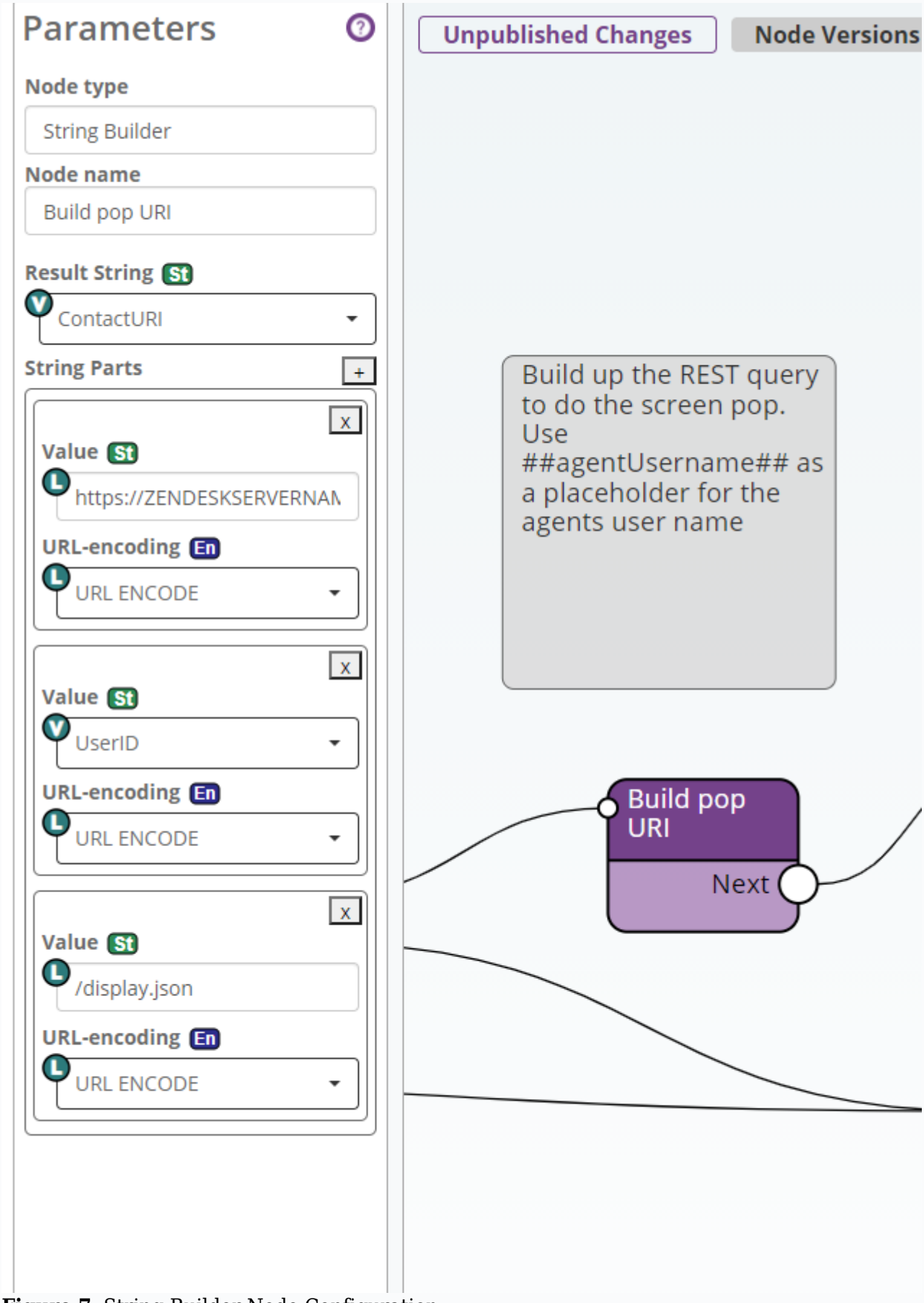

Figure 7. String Builder Node Configuration

Use a Prepare REST Request to Set Up the Screen Pop to Fire When the Agent’s Phone is Ringing

The Prepare REST node enables us to set up a REST request to be run on an event. In this case, it is set to trigger when a call leaves the queue and the agent’s phone is ringing.

We use the ContactURI variable set in the previous step for the Rest request. The Auth token is put in the header as per the previous request.

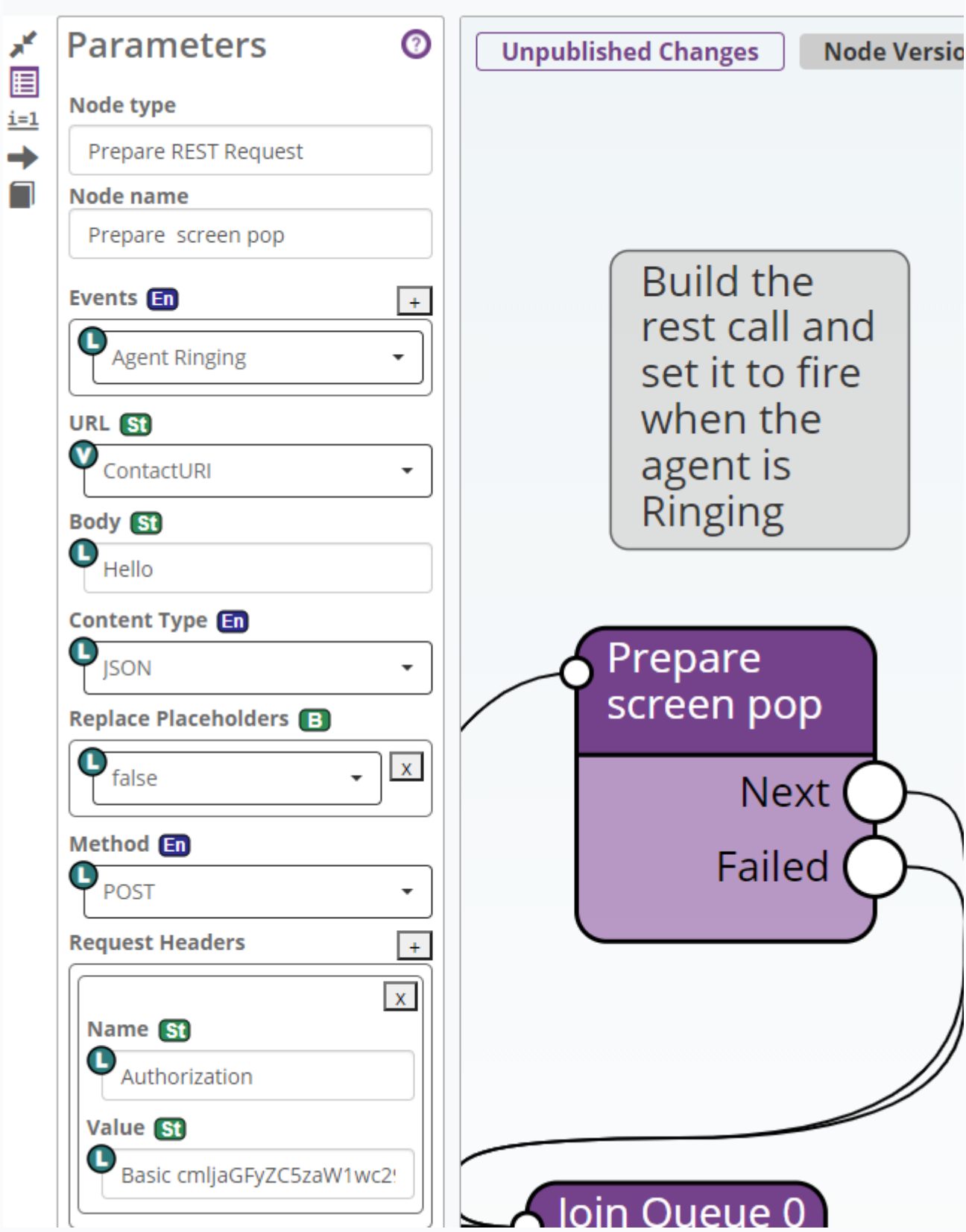

Figure 8. Prepare REST Request node configuration

Call is Put Into The Queue Ready to be Assigned to an Agent

The call can now be placed in a queue. If the contact's number is associated with a customer record in Zendesk, the REST query will run as the agent's phone starts to ring, causing Zendesk to display the appropriate customer record.

Nodes That Can be Used to Interact With a Third-Party Application

Inbound Call

This node type is usually the first node in a call flow. It is the starting point of the call flow and allows you to access call-leg information.

The following parameters can be set:

Parameter		How many?	Type	Description
Options				Optional parameters for the inbound call.
	Inbound Call	0 or 1	CallLeg	An optional variable in which to store the inbound call leg (the A leg) so that it can be accessed later on in the call flow.
	Call ID	0 or 1	String	An optional variable in which to store the call identifier so that it can be accessed later on in the call flow.
	Calling Number	0 or 1	String	An optional variable in which to store the calling number so that it can be accessed later on in the call flow. If the CLI is restricted, this is set to the value CLIR.
	CLI Restricted	0 or 1	Boolean	An optional variable that determines whether or not the CLI is restricted.
	Service Number	0 or 1	String	An optional variable in which to store the service number so that it can be accessed later on in the call flow.
	Service Number Name	0 or 1	String	An optional variable in which to store the service number name so that it can be accessed later on in the call flow.
	Overdial	0 or 1	String	An optional variable in which to store overdialled numbers (the additional digits dialled on top of a service number).

Parameter		How many?	Type	Description
	Auto Answer	0 or 1	Boolean	By default, calls are automatically answered immediately (this parameter is set to True). If auto answering is disabled (set to False) or not set at all, calls can then be answered when one of the following applies: The Contact Centre is specifically instructed to do so (with an AnswerCall node in the call flow) The called party answers the call Prior to caller interaction (DTMF keypresses or speech-recognition). Note: If required, this parameter can be set to User-configurable so that it can accept a value specified by a company administrator. For more information, see Adding user-configurable values.
	Connect Media	0 or 1	Boolean	An optional setting that determines whether or not the Contact Centre attempts to establish early media to the calling party. If this parameter is absent or is set to True (the default setting), the Contact Centre immediately attempts to establish early media to the calling party. To prevent this, set the parameter to False. Note: If required, this parameter can be set to User-configurable so that it can accept a value specified by a company administrator. For more information, see Adding user-configurable values.

Chat Arrived

This node type is the starting point of the chat flow. It executes either when a chat message arrives or when the presence of a customer is detected.

The following parameter can be set:

Parameter		How many?	Type	Description
Options				Optional parameters.
	Chat ID	0 or 1	String	An optional variable in which to store the chat identifier so that it can be accessed later on in the chat flow.

The following branches can be taken:

Branch	How many?	Description
Message Arrived	0 or 1	The branch to take when a chat message arrives.

Branch	How many?	Description
Presence Arrived	0 or 1	The branch to take when the presence of a customer is detected, such as when they open the chat panel. If this branch is configured, the chat flow waits for a given time to receive a chat message. If it does not receive one, it sends a “Hi. Can I help you?” message. If this branch is missing, the incoming presence is ignored and the chat flow waits to receive a chat message before moving on to the next node in the routing flow. Note: If both the Chat Arrived branch and the Presence Arrived branch are configured, the node takes the branch that is matched first.

String Builder

This node type concatenates a set of strings and integers.

The resulting string variable contains input values from call-flow variables that have either been defined as parameters for previous nodes in the call flow or added as new call flow variables for this node type. If required, each value can be URL-encoded.

The following parameters can be set:

Parameter		How many?	Type	Description
Result String		1	String	The string variable to receive the result. This is the final output string.
String Parts		0 to 30		
	Value		String	The string or integer value to assign to the string part. Note: If you set a string or an integer value for previous nodes in the call flow, they will be enlisted and used as inputs in this node too. However, if someone tries to create a variable in this node (other than that of the final output string), it will be of type string (not integer). Note: If required, this parameter can be set to User-configurable so that it can accept a value specified by a company administrator. For more information, see Adding user-configurable values.
	URL Encoding		Enum	Either NONE or URL ENCODE.

The following branch can be taken:

Branch	How many?	Description
Next	1	This branch is always taken.

REST Query

This node type is used to send an arbitrary HTTP request to an external HTTP server. It might be used, for example, to retrieve details from an external CRM system or to inform an external system that a call has arrived.

This node allows, for example, external systems to be notified of or passed information pertaining to Contact Centre calls, such as individual customer interactions, statistics, etc. for monitoring, configuration and data-analysis purposes.

If the Replace Placeholders parameter is set to true, the system will replace any placeholders found in the body text with the appropriate values when the request is sent. Supported placeholders are: ##agentExternalId##, ##agentUsername##, ##event##, ##queueName## and ##time##. For convenience, a Simple JSON node can be used to construct a JSON string that includes these placeholders as values. These placeholders will be replaced as follows:

##agentExternalId## – replaced with the External ID value as configured for an agent's user account, in the Administrator Portal.

##agentUsername## – replaced with the username of the agent who is being called

##event## – replaced with one of the following strings: AGENT_RINGING, AGENT_ANSWERED, TRANSFER_RINGING, or TRANSFER_ANSWERED

##queueName## – replaced with the name of the queue on which the current call resides or, in the case of an attended queue consult/transfer, the name of the target queue

##time## – replaced with a string in ISO 8601 compatible format.

If required, you can modify the header when sending a HTTP request (for example, for authentication purposes) and add additional headers. If the response to the REST request contains a JSON body, the node type allows for specified parts of that data to be read out into provided variables.

Note: Unlike the Prepare REST Request node, this node type sends the HTTP request immediately.

Note: If required, some of the parameters for this node type can be set to User-configurable so that they can accept values specified by a company administrator. For more information, see Adding user-configurable values.

The following parameters can be set:

Parameter	How many?	Type	Description
URL	1	String	The URL for this HTTP request. This parameter can be set to User-configurable.
Body	1	String	The body of the outbound HTTP request. This parameter can be set to User-configurable.

Parameter			How many?	Type	Description
Content Type			1	Enum	One of: JSON, TEXT/PLAIN or TEXT/HTML.
Replace Placeholders			0 or 1	Boolean	Whether or not placeholder tags are replaced in the body of the HTTP request at the point when the HTTP request is sent. The placeholders are defined above. This parameter can be set to User-configurable.
Method			1	Enum	The HTTP method to use for the request. One of: POST, GET, DELETE or PUT. Note: If the HTTP method is set to GET, the Body parameter is omitted.
Request Headers				0 to 50	Optional headers for the HTTP request.
	Name		0 or 1	String	The name of the header in the HTTP request. This parameter can be set to User-configurable.
	Value		0 or 1	String	The value to assign to the header. This parameter can be set to User-configurable.
Response Data					Optional values for different parameter types that can be set from a JSON response body. If the HTTP response contains a JSON body, these optional parameters allow parts of that JSON data to be read into the provided variables.
	Integer Parameters		0 to 50		
		Variable	0 or 1	Integer	The variable into which to place the read value.
		Path	0 or 1	String	The path of the value to read. See the example below. This parameter can be set to User-configurable.
		Default	0 or 1	Integer	The value to write to the variable if no value is found at the given path. This parameter can be set to User-configurable.
	Float Parameters		0 to 50		

Parameter			How many?	Type	Description
		Variable	0 or 1	Float	The variable into which to place the read value.
		Path	0 or 1	String	The path of the value to read. See the example below. This parameter can be set to User-configurable.
		Default	0 or 1	Float	The value to write to the variable if no value is found at the given path. This parameter can be set to User-configurable.
	String Parameters		0 to 50		
		Variable	0 or 1	String	The variable into which to place the read value.
		Path	0 or 1	String	The path of the value to read. See the example below. This parameter can be set to User-configurable.
		Default	0 or 1	String	The value to write to the variable if no value is found at the given path. This parameter can be set to User-configurable.
	Boolean Parameters		0 to 50		
		Variable	0 or 1	Boolean	The variable into which to place the read value.
		Path	0 or 1	String	The path of the value to read. See the example below. This parameter can be set to User-configurable.
		Default	0 or 1	Boolean	The value to write to the variable if no value is found at the given path. This parameter can be set to User-configurable.

Note: If the external HTTP server does not return a value or if the type of returned value is incompatible with the call-flow variable type, (for example, the server returns the string “hello” in the response body in a Path that is assigned to a Boolean parameter, the Default parameter value is used and the Next branch is taken.

The following branches can be taken:

Branch	How many?	Description
Next	1	The branch to take if the HTTP request succeeds.
Failed	1	The branch to take if the HTTP request fails. Failure is indicated by any response code in the range 400 to 599.

Example Usage of the JSON Path

An HTTP REST request is sent to an external HTTP server, which returns the following JSON:

```
{
    "result": {
      "success": true,
      "details": {
        "code": 100,
        "message": "Success"
      }
    },
    "data": [
      { "name": "Alice", "age": 27 },
      { "name": "Bob", "age": 47 }
    ]
  }
```

Given the above data, the following paths would result in the following values being written to the provided variable:

Path	Type	Value
result.success	Boolean	true
result.details.message	String	Success
data.0.name	String	Alice
Data.1.age	Integer	47

This data is then passed to a Compare String node, which directs the call as appropriate.

Compare Integer

This node type allows branching based on an integer value.

To branch on a specific integer value, you need to add an optional branch by right-clicking on the node name and selecting Add “Match” Branch.

To set the branch parameters (branch name, match type and match value), select the new branch by left-clicking on the branch.

The following parameter can be set:

Parameter	How many?	Type	Description
Value	1	Integer	(Mandatory) An integer value to match. This is usually a variable. Note: If required, this parameter can be set to User-configurable so that it can accept a value specified by a company administrator. For more information, see Adding user-configurable values.

The following branches can be taken:

Branch		How many?	Type	Description
Match		1 to 50		The branch to take if a specific integer is matched.
	Match Type		Enum	The type of match: Equal to, Less than or Greater than.
	Match Value		Integer	The integer to match. Note: If required, this parameter can be set to User-configurable so that it can accept a value specified by a company administrator. For more information, see Adding user-configurable values.
No Match		1		The branch to take if no other branches match.

Prepare REST Request

This node type is used to prepare an arbitrary HTTP request to be sent to an external platform. Any prepared requests will be sent when the specified agent-related events occur.

The HTTP request is only sent when triggered by a specific event, such as when an agent is being alerted to an incoming call or when they answer a call. Multiple REST queries can be prepared for the same event.

This node allows, for example, external systems to be notified of or passed information pertaining

to Contact Centre calls, such as individual customer interactions, statistics, etc. for monitoring, configuration and data-analysis purposes.

If the Replace Placeholders parameter is set to true, the system will replace any placeholders found in the body text with the appropriate values when the request is sent. Supported placeholders are: ##agentExternalId##, ##agentUsername##, ##event##, ##queueName## and ##time##. For convenience, a Simple JSON node can be used to construct a JSON string that includes these placeholders as values. These placeholders will be replaced as follows:

##agentExternalId## – replaced with the External ID value as configured for an agent's user account, in the Administrator Portal.

##agentUsername## – replaced with the username of the agent who is being called

##event## – replaced with one of the following strings: AGENT_RINGING, AGENT_ANSWERED, TRANSFER_RINGING, or TRANSFER_ANSWERED

##queueName## – replaced with the name of the queue on which the current call resides or, in the case of an attended queue consult/transfer, the name of the target queue

##time## – replaced with a string in ISO 8601 compatible format.

If required, you can modify the header when sending a HTTP request (for example, for authentication purposes) and add additional headers.

Note: Unlike the REST Query node, this node type does not send the HTTP request immediately, but instead stores the request details so that the request can be sent when a particular event (or a set of events) occurs.

Note: If required, some of the parameters for this node type can be set to User-configurable so that they can accept values specified by a company administrator. For more information, see Adding user-configurable values.

The following parameters can be set:

Parameter	How many?	Type	Description
Events	0 to 4	Enum	When this HTTP request should be sent: Agent Ringing – As soon as the agent's phone starts to ring Agent Answered – As soon as the agent answers a call Transfer Ringing – As soon as the phone starts to ring for a consulted party or one to which a call has been transferred Transfer Answered – As soon as the consulted party or one to which a call has been transferred answers the call
URL	1	String	The URL for this HTTP request. This parameter can be set to User-configurable.

Parameter	How many?	Type	Description
Body	1	String	The body of the outbound HTTP request. This parameter can be set to User-configurable.
Content Type	1	Enum	One of: JSON, TEXT/PLAIN or TEXT/HTML.
Replace Placeholders	0 or 1	Boolean	Whether or not placeholder tags are replaced in the body of the HTTP request at the point when the HTTP request is sent. The placeholders are defined above. This parameter can be set to User-configurable.
Method	1	Enum	The HTTP method to use for the request. One of: POST, GET, DELETE or PUT. Note: If the HTTP method is set to GET, the Body parameter is omitted.
Request Headers		0 to 50	Optional headers for the HTTP request.
Name	0 or 1	String	The name of the header in the HTTP request. This parameter can be set to User-configurable.
Value	0 or 1	String	The value to assign to the header. This parameter can be set to User-configurable.

The following branches can be taken:

Branch	How many?	Description
Next	1	The branch to take if the HTTP request succeeds.
Failed	1	The branch to take if the HTTP request fails. Failure is indicated by any response code in the range 400 to 599.