Zendesk Service Desk Integration

Version: 8.2 Patch 2

Copyright and Trademark Notices

Contents

IdentityIQ for Zendesk Service Desk

The following topics are discussed in this guide:

Overview

The Rest based integration between SailPoint and IdentityIQ for Zendesk Service Desk system enables customers to create incident type of ticket in IdentityIQ for Zendesk Service Desk system for the configured operations (for example, creating account, removing /deleting access and other operations) using rest APIs. The seamless integration of SailPoint IdentityIQ for Zendesk Service Desk system eliminates the need to build and maintain a custom integration and reduces time-to-deployment.

> The Zendesk Service Desk must be configured using backend and would require SailPoint Services hours. For more information, contact the SailPoint Customer Success Manager.

Supported Features

IdentityIQ for Zendesk Service Desk supports the following features:

- Creating ticket of types Incident for all account provisioning.
- Syncing ticket status between the two systems.
- Retry Mechanism for Create Ticket request failure.

Supported Managed Systems

IdentityIQ for Zendesk Service Desk supports the following versions of the Zendesk product:

- Zendesk for service (cloud)

Prerequisites

- *For Basic Authentication*: Ensure that you set the **Password Access** or **Token Access** as **Enabled** in the Zendesk API admin configuration.
- *For OAuth 2.0 Authentication*:

- Ensure that you set the **Password Access** as **Enabled** in the Zendesk API admin configuration.
- Create a new OAuth client by providing the **Client Name** in the Zendesk API admin configuration. Use the same **Client Name** for the **Client ID** and generated **Client Secret** on your Zendesk application.

> **Client Secret** is only visible once in the Zendesk managed system, and while creating the OAuth Client.

- Aggregate the users from Zendesk system into IdentityIQ.

Required Permissions

Assign the following minimum access permission to create and get status of ticket:

- Agent role is required for service account.

Configuring IdentityIQ to Integrate with IdentityIQ for Zendesk Service Desk

This section provides the required information for configuring IdentityIQ to integrate with Zendesk Service Desk.

This is intended as an introduction to the configuration required to integrate IdentityIQ with Zendesk Service Desk. It outlines some examples that must be used as a reference point for implementation. Some changes may be required to meet specific use case and expertise around both systems are a must for the successful implementations.

SailPoint provides a default Zendesk Service Desk configuration. This configuration implements the integration between IdentityIQ and the Zendesk Service Desk to fulfil (fulfilment of the ticket is done manually) creation of tickets based on IdentityIQ access certification remediation events.

The default configuration is located in the following directory, where *iiqHome* is the location where IdentityIQ was installed:

```
iiqHome/WEB-INF/config/connector/IdentityIQforZendeskServiceDesk.xml
```

> Once the following configuration information is populated then import the `IdentityIQforZendeskServiceDesk.xml` file. This would create an application.

The configuration must include the following entries:

*url**

The base URL of Service Desk System based on the **authenticationType** as follows:

- **Basic**: https://{subdomain}.zendesk.com

*authenticationType**

Authentication method that is supported by the managed system:

- Basic
- OAuth2

To use an API token with Basic Authentication, append the username with **/token** and provide the API token in the **Password** field.

For more information on creating the API Token, see Generating an API Token.

requesterSource

The application name by which Zendesk Service Desk accounts are aggregated. Required for Plan Initializer script.

ticketType

Enter one of the following ticket type to generate ticket on Zendesk Service Desk system:

- incident

Applicable if authenticationType is selected as Basic

*username**

Service Account username.

If API Token is used then append the username with **/token** Service Account user’s Password or API Token.

Applicable if authenticationType is selected as OAuth2

token_url

URL for generating refresh token.

For example, **https://{subdomain}.zendesk.com/oauth/tokens**

grant_type

Enter the following type of Grant:

PASSWORD

client_id

Client Id for OAuth2 authentication.

client_secret

Client secret for OAuth2 authentication.

Application Configuration XML would have all configurations for Incident. This configuration would be executed by connector for the request processing.

*scope**

Required. A space-separated list of scopes that control access to the Zendesk resources.

*username**

(*Applicable if grant_type is selected as PASSWORD*) Service Account username.

*password**

(*Applicable if grant_type is selected as PASSWORD*) Service Account user password.

Each module would have **provision** and **checkStatus** entries as mentioned below:

Provision:

Entries	Description
resource	Ticket creation rest endpoint. Do not provide the base url in the value. Base url would

Entries	Description
	be appended to this endpoint value. Provide only remaining endpoint URL. IdentityIQ For Zendesk Service Desk: **/api/v2/tickets**
responseElement*	The value is JSON path expression which provides information about where to find ticket number in the response from rest endpoint. For example, **$.ticket.id**
request*	Map that represents the request payload, which has velocity template expression and velocity variables that would be dynamically updated by integration before making rest call.
requestRootElement	The value represents JSON root element in the request.
requestRootElementType	The value represents JSON root element's type in the request. For example, JSONObject
requester_id*	The value represents the id of the reporter.
submitter_id	The value represents the id of the reporter. Provide **id** of service account user, if you wish to populate submitter as service account. By default it will be same as requester.
comment*	The map of comment filed values containing description of the ticket.
body*	The main body of the request in Zendesk Service Desk, which includes the details about the request.
subject*	Title / Summary of the ticket in Zendesk.
type*	Ticket type in Zendesk system.

Check Status:

Entries	Description
resource	Ticket creation rest endpoint. Do not provide the base url in the value. Base url would be appended to this endpoint value. Provide only remaining endpoint URL. IdentityIQ For Zendesk Service Desk: **/api/v2/tickets/$ticketId**
responseElement*	The value is JSON path expression which provides information about where to find ticket number status in the response from rest endpoint. For example, **$.ticket.status**
statusMap	Map that relates Ticket System status to IdentityIQ status.

If any changes required in the mapping, change the default value /key values in **statusMap** as mentioned in the following tables:

statusMap for Incident

Entry key (Zendesk) Status	Values (IdentityIQ)
new	Queued
open	Queued
pending	Queued
solved	Committed
closed	Committed

Retryable Mechanism

For availing the advantage of some of the logic around retryable situations, add the retryable error messages list to the attributes map on an application. The **retryableErrors** entry is a list of strings through which the connector searches when it receives a message from the managed application. If one of the strings in the entry exists in the error, the connector attempts to retry the connection. When the configured error string is not a part of the error message returned from the connector, then IdentityIQ would not attempt a retry.

For example:

```
<entry key="retryableErrors">
    <value>
      <List>
        <String>Connection reset</String>
      </List>
    </value>
</entry>
```

Enter the following command to enable **log4j2** logging on Service Desk component:

```
logger.ZendeskSDIM.name=openconnector.connector.servicedesk.ServiceDeskConnector
```

```
logger.ZendeskSDIM.level=debug,file
```

Creating Multiple Tickets in Zendesk Service Desk System

To create multiple tickets in the Zendesk Service Desk System using IdentityIQ, add the following attributes in the IdentityIQforZendeskServiceDesk.xml file:

- `multipleTicket`: You can specify the following values:
    - `true`: A separate Zendesk Service Desk ticket is created for each line item from the IdentityIQ access request.
    - `false` (this is the default): A single Zendesk Service Desk ticket is created against all line items from the IdentityIQ access request.

      The entries use the following format:

      ```
      <entry key='multipleTicket' value='false'/>
      ```

- `groupTicketBy`: You can specify the following values:
  - `none`: If the attribute is not defined or if the attribute value is other than `Application`, then IdentityIQ sets this attribute to `none`.
  - `Application` (this is the default): If the attribute value is `Application` and `multipleTicket=true`, then IdentityIQ access request lines from the same application are moved to a single ticket.

    The entries use the following format:

    ```
    <entry key='groupTicketBy' value='Application'/>
    ```

For example, you can enter the `multipleTicket` and `groupTicketBy` keys in the integration configuration file as follows:

```
<entry key='multipleTicket' value='true'/>
<entry key='groupTicketBy' value='Application'/>
```

Troubleshooting

Ticket creation fails because the request includes escape characters (\\)

Ticket creation fails for request having escape characters (\\) with the following error message:

```
Ticket creation failed. com.google.gson.stream.MalformedJsonException: Invalid
escape sequence at line 3 column 49 path $.body
```

Resolution: Edit the Body attribute by replacing the existing text with:

```
#foreach($req in $plan.requests) #if($req.operation == 'Create') Create Account
on application $req.resource #else For $req.id in application $req.resource #end
#if($req.items) $newline #foreach($item in $req.items) #if ($item.name ==
'*disabled*' &amp;&amp; $item.value == 'true') Disable Account. $newline #elseif
($item.name == '*disabled*' &amp;&amp; $item.value == 'false') Enable Account.
$newline #elseif ($item.name == '*locked*' &amp;&amp; $item.value == 'false')
Unlock Account. $newline #else $!item.Operation $item.name:
$$StringEscapeUtils.escapeJava ($StringEscapeUtils.escapeJava
($item.value.toString())) $newline #end #end #else $newline $!req.Operation
Account #end $newline #end" />
```