OpenText™ Identity Governance Collectors and Fulfillers

Configuration Guide

25.1 (v4.5)

Legal notice

Contents

About this book and the library

The 24.4 (v4.4) *OpenText Identity Governance Collectors and Fulfillers Configuration Guide* provides information about the various collectors and fulfillers supported by OpenText Identity Governance and how they are integrated with different applications to simplify the process of collection and fulfillment.

Intended audience

This guide provides information for OpenText Identity Governance Data Administrator and others responsible for configuring collection and fulfillment target templates.

Other information in the library

The library provides the following information resources in addition to this guide. Visit the Documentation Web site (https://www.microfocus.com/documentation/identity-governance-and-administration/) to access these and other documents in this library.

Release Notes

Provides information specific to this release of the OpenText Identity Governance product, such as known issues.

User and Administration Guide

The User and Administration Guide provides a step-by-step guidance for the OpenText Identity Governance user-oriented and administration tasks, specifically, for the following users:

- Access requesters
- Access Request approvers
- Reviewers
- Review owners
- Fulfillers
- Application owners
- Separation of Duties Policy owners
- Business Role managers
- All administrator authorizations

Reporting Guide

Provides information about Identity Reporting for OpenText Identity Governance and how you can use the features it offers.

Technical References

Provide specific details about narrow topics relevant to few use cases.

1 Introduction

OpenText Identity Governance collects information from various identity and application data sources including OpenText™ Identity Manager and guides **authorized users** through the key phases of the governance process. You can install OpenText Identity Governance on premises or access it as a service, and then collect data from cloud and on-premises data sources and send change requests to fulfillment targets. When using OpenText Identity Governance as a service and collecting data from on-premises systems, you will need to install the Cloud Bridge Agent and configure data source connections in OpenText Identity Governance.

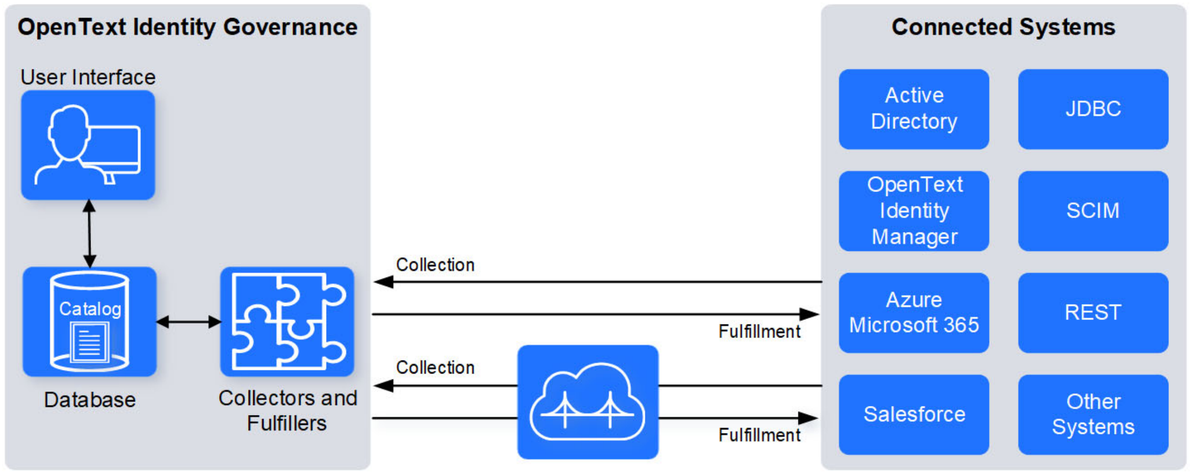


Figure 1-1 *Collection and fulfillment Overview*

For an overview about integrating with data sources and fulfillment targets, templates, and authentication methods, see the following sections:

- Section 1.1, "Integration with data sources and fulfillment targets," on page 9
- Section 1.2, "Collector and fulfiller templates," on page 10
- Section 1.3, "Authentication methods," on page 10

1.1 Integration with data sources and fulfillment targets

OpenText Identity Governance enables you to collect (pull) data from connected systems, transform that data and perform governance tasks, and send change requests (push data) to connected systems to ensure that the right people have the right access.

Businesses can authorize administrators to perform integration activities. However, we strongly recommend that you create a dedicated account for the integration activities. This would ensure that integration authorization is not associated with a specific user but associated with an account for business continuity and auditing purposes.

Based on the desired integration, specific capabilities for the integration account user might be required. For guidance related to procedures for creating the integration account, refer to the documentation of the systems that you want to integrate with OpenText Identity Governance. For example, you can refer to the Salesforce knowledge article on the Salesforce website to create an integration account for Salesforce. For a Workday integration account, refer to your Workday documentation.

1.2 Collector and fulfiller templates

OpenText Identity Governance provides templates that enable you to easily integrate with LDAP systems, service desk systems, protocols, and specific applications, to collect data and fulfill change requests. The **collectors** enable you to collect identities, applications, accounts, and permissions to provide a view of all your enterprise data in the OpenText Identity Governance catalog. OpenText Identity Governance collects the attributes configured in the collector templates and those are mapped to OpenText Identity Governance attributes. The **fulfillment targets (fulfillers)** enable you to fulfill change requests generated from reviews, requests, and catalog curation. The fulfiller templates also include default mappings. You can edit the provided templates and add attributes to the template. The templates also include transformation scripts that you can edit as needed.

For basic template layout overview, change event processing, and configuration procedures common to multiple collectors and fulfillment targets, refer to the following sections and chapters of the *OpenText Identity Governance as a Service User and Administration Guide: (https://www.microfocus.com/documentation/identity-governance-and-administration/igaas/user-guide/front.html)*:

- Understanding collectors
- Collecting identities
- Collecting applications and application data
- Configuring fulfillment

Many of these templates are easy to use with tooltips and preconfigured default values. A few templates for complex systems might need additional guidance because of the variations in required minimum rights, authentication methods, identity categorizations, parent-child relationships, and other such unique features of the connected systems. This guide provides the additional guidance required to configure collectors and fulfillment targets successfully.

1.3 Authentication methods

By default, many of the templates use the Basic Authentication method which requires a user name and password to connect with other applications. When using Cloud Bridge, you will need to specify 0 as the ordinal for the basic authentication method. For templates such as Identity Manager AE Permission Collector, SCIM collector, or Workday fulfillment target, you might specify other authentication methods. For information about the other authentication methods, see the connector-specific chapters in this guide. As stated earlier, this guide provides additional guidance required to configure collectors and fulfillment targets successfully. For basic collection and fulfillment procedures, see the *OpenText Identity Governance as a Service User and Administration Guide (https://www.microfocus.com/documentation/identity-governance-and-administration/igaas/user-guide/front.html)* .

For information about Cloud Bridge Agent installation and administration, see the *OpenText Cloud Bridge Installation and Administration Guide (https://www.microfocus.com/documentation/identity-and-access-management/iam-services/cloud-bridge-agent-admin/bookinfo.html)*.

2 List of collector and fulfillment target templates

OpenText Identity Governance provides the following templates.

IMPORTANT: Work with your integration account and network administrators to ensure that you have the minimum rights to the connected systems and that your system has the required security certificates. After initial configuration, you must always update credentials and other service parameters in each template as needed. For example, when connecting to applications that use access tokens for authentication, such as SCIM-compatible applications, you must change tokens when they expire, then reconfigure the template to use the current access token.

Data Sources and Fulfillment Systems	Identity Collector Templates	Account Collector Templates	Permission Collector Templates	Fulfillment Target Templates
Active Directory	◆ AD Identity ◆ AD Identity with Changes	◆ AD Account	◆ AD Permission ◆ AD Hybrid Permission	◆ Active Directory LDAP Fulfillment
Azure AD	◆ Azure AD MS Graph Identity	◆ Azure AD MS Graph Account	◆ Azure AD MS Graph Permission ◆ Azure AD MS Graph Service Plan	◆ MS Azure AD Fulfillment
CSV	◆ CSV Identity	◆ CSV Account	◆ CSV Permission	◆ CSV Fufillment
eDirectory	◆ eDirectory Identity ◆ eDirectory Identity (W/O IDM) with Changes	◆ eDirectory Account	◆ eDirectory Hybrid Permission ◆ eDirectory Permission	◆ eDirectory LDAP Fufillment
Google Apps	◆ Google Apps User Identity	◆ Google Apps User Account	◆ Google Apps User Permission	

Data Sources and Fulfillment Systems	Identity Collector Templates	Account Collector Templates	Permission Collector Templates	Fulfillment Target Templates
Identity Manager	• Identity Manager Identity • IDM with Changes	• IDM Entitlement Account	• Identity Manager AE Permission (only for Identity Manager AE systems) • IDM Entitlement Permission	• Identity Manager Dxcmd Fulfillment for Active Directory • Identity Manager Automated (system) (only for Identity Manager AE systems) • IDM Entitlement Fulfillment
ILM	• ILM Identity	• ILM Account	• ILM Permission	• ILM Fulfillment
JDBC	• JDBC Identity	• JDBC DB2 Account • JDBC Generic Account • JDBC MySQL Account • JDBC Non-specific Account • JDBC Oracle Account • JDBC PostgreSQL Account • JDBC SQL Server Account • JDBC Sybase Account	• JDBC DB2 Permission • JDBC Generic Permission • JDBC MySQL Permission • JDBC Non-specific Permission • JDBC Oracle Permission • JDBC PostgreSQL Permission • JDBC SQL Server Permission • JDBC Sybase Permission	• JDBC Generic DB Fulfillmemt • JDBC Oracle Fulfillment • JDBC PostgreSQL Fulfillemt • JDBC SQL Server Fulfillment
MS Teams			• MS Teams Permission	• MS Teams Fulfillment
RACF	• RACF Identity	• RACF Account	• RACF Permission	

Data Sources and Fulfillment Systems	Identity Collector Templates	Account Collector Templates	Permission Collector Templates	Fulfillment Target Templates
REST GitHub		• REST GitHub Account	• REST GitHub Organization Permission • REST GitHub Repository Permission • REST GitHub Team Permission	• REST GitHub Fulfillment
REST OTDS	REST OTDS Identity	REST OTDS Account	• REST OTDS Hybrid Permission • REST OTDS Permission	• REST OTDS Fulfillment
REST PAM		• REST PAM Account	• REST PAM Permission	
Salesforce	• Salesforce Identity	• Salesforce Account	• Salesforce Permission • Salesforce Profile Permission • Salesforce Role Permission	• Salesforce Fulfillment
SAP (Not supported in OpenText Identity Governance as a service environments)	• SAP HR Identity • SAP User Management Identity	• SAP User Management Account	• SAP User Management Permission	
SCIM	• SCIM Identity	• SCIM Account	• SCIM Permission	• SCIM Fulfillment
ServiceNow	• ServiceNow Identity	• ServiceNow Account	• ServiceNow Permission	• ServiceNow Generic Fulfillment • ServiceNow Incident Fulfillment • ServiceNow Request Fulfillment • ServiceNow Task Fulfillment
Sharepoint	• Sharepoint Identity	• Sharepoint Account		

Data Sources and Fulfillment Systems	Identity Collector Templates	Account Collector Templates	Permission Collector Templates	Fulfillment Target Templates
Workday	◆ Workday Identity	◆ Workday Account	◆ Workday Permission	
All systems				◆ Manual Fulfillment (system default) ◆ Identity Manager workflow (system) ◆ CSV Fulfillment ◆ REST Generic ◆ SCIM Fulfillment ◆ Workflow Service
Any system reachable via http/ https (excluding systems that use SOAP and REST service)				◆ Generic Http Fulfillment
BMC Remedy systems				◆ BMC Remedy Incident Fulfillment (Deprecated)
SOAP systems				◆ SOAP Service Fulfillment

3 Active Directory and OpenText eDirectory collectors and fulfillers

This chapter focuses on specific configuration-related information regarding the following collection and fulfillment templates:

- AD Identity
- AD Identity with changes
- eDirectory Identity
- eDirectory Identity (w/o IDM) with changes
- eDirectory Hybrid permission
- eDirectory Account
- eDirectory Permission
- AD Account
- AD Permission
- AD Hybrid permission
- Active Directory LDAP Fulfillment
- eDirectory LDAP Fulfillment

For information about configuring AD and OpenText™ eDirectory templates, see the following sections:

- Section 3.1, "AD and OpenText eDirectory collectors," on page 17
- Section 3.2, "Active Directory and OpenText eDirectory LDAP fulfillment," on page 18

3.1 AD and OpenText eDirectory collectors

To ensure synchronization of data from OpenText eDirectory to the OpenText Identity Governance catalog, the users or groups in OpenText eDirectory must have the required minimum rights in the OpenText eDirectory repository. The following rights are required for data synchronization:

- For full synchronization: Read permission on the users and their attributes that are collected
- For fast synchronization: Read permission on the users and their attributes that are collected
- For fulfillment: Read and write permission on the users and their attributes for whom the fulfillment request is raised

The OpenText Identity Governance collectors for OpenText eDirectory have two identity collector templates. The **eDirectory Identity** template is used when the connected system has both OpenText eDirectory and OpenText Identity Manager installed, whereas the **eDirectory Identity > (w/o IDM)**

with changes template is used when the connected system has OpenText eDirectory installed with the change-log module. The change-log module enables the connector to recognize the changes that require publication from the connected system to the OpenText Identity Governance catalog.

For more information about collecting identities with changes and the change event collection, and for more information about applying changes see "Collecting from Identity Sources with Change Events (https://www.microfocus.com/documentation/identity-governance-and-administration/igaas/user-guide/t4ndywshnp9q.html)" and "Understanding Change Event Processing (https://www.microfocus.com/documentation/identity-governance-and-administration/igaas/user-guide/t4gd7f44rwe5.html)" in the *OpenText Identity Governance as a Service User and Administration Guide (https://www.microfocus.com/documentation/identity-governance-and-administration/igaas/user-guide/front.html)*.

For OpenText Identity Governance to associate the accounts and permissions with the identities available in the catalog, while configuring the template, in the **Collect Account** view, use `mail` as the **Account-User Mapping** attribute and `email` as the **Map to attribute**. In the **Collect Permission** view, use `member` as the **Permission-Account or User Mapping** attribute and `Account ID from Source` as the **Map to attribute**.

In addition to these collectors there are the OpenText eDirectory and AD hybrid collectors for collecting permissions. For more information about hybrid collectors, see "Understanding Hybrid Permission Collectors (https://www.microfocus.com/documentation/identity-governance-and-administration/igaas/user-guide/t4kwieos9qlg.html)" in the *OpenText Identity Governance as a Service User and Administration Guide (https://www.microfocus.com/documentation/identity-governance-and-administration/igaas/user-guide/front.html)*.

3.2 Active Directory and OpenText eDirectory LDAP fulfillment

If a user is present in OpenText Identity Governance but is not present in either Active Directory or OpenText eDirectory, you can configure the fulfillment target to create an account through the respective fulfillment targets.

NOTE: Before you configure a fulfillment target with either an Active Directory LDAP fulfillment type or an OpenText eDirectory LDAP fulfillment type, you must ensure that Active Directory collects the attributes required for fulfillment. To verify Active Directory or OpenText eDirectory LDAP collection, log in to OpenText Identity Governance and then click **Data Sources** > **Application Definition Sources**.

To configure the fulfillment target, in Step 4b, you must provide values for the **first name**, **last name**, **title**, and **workforceID** fields.

In addition, when you configure **Fulfillment item configuration and mapping**, click **{...}**, then edit the transform script for the **Account name generation payload** to connect to the correct Active Directory or OpenText eDirectory server for the user.

4 Azure AD MS graph collectors and fulfiller

This chapter focuses on specific configuration-related information regarding the following collection and fulfillment templates:

- Azure AD MS Graph Identity
- Azure AD MS Graph Account
- Azure AD MS Graph Permission
- Azure AD MS Graph Service Plan
- MS Azure AD Fulfillment

For information about configuring Azure AD MS Graph templates, see the following sections:

- Section 4.1, "Azure AD collectors," on page 19
- Section 4.2, "Azure AD MS graph fulfillment," on page 21

4.1 Azure AD collectors

When your environment uses both Active Directory and Azure AD, user identities might be unique to one of the applications or might exist in both applications. If you use Active Directory and Azure AD with DirSync or AD Connect, you can create a single identity source for both applications by using the Azure AD User collector template.

In the collector template, specify an attribute that you want to use for merging duplicate identities and for matching identities to accounts and permissions. The attribute for the matching rule should contain a value that is unique to each identity. For example, in AD and OpenText Identity Manager, each user tends to have a unique `Distinguished Name`.

IMPORTANT: We have deprecated the 3.6.2 Azure AD User templates because Azure AD Graph is no longer supported by Microsoft. If you are still using the old Azure AD templates, you can reconfigure your template to map to the Microsoft Graph API by changing the **Azure AD Service Resource** default value to `https://graph.microsoft.com/v1.0`. For information about the differences between the previously supported API and Microsoft Graph API, see Microsoft's Learning Portal.

When using the Azure AD MS Graph collector, complete the following steps :

1. Enable the Azure Microsoft Graph API for your site and grant the following permissions, ranging from the least to the most privilege, to an account to access the API:

Permission	Types	Description
`Application.Read.All`	Application	Read all applications
`Device.Read.All`	Application	Read all devices
`Directory.AccessAsUser.All`	Delegated	Access the directory as the signed-in user
`Directory.Read.All`	Application and Delegated	Read directory data
`Directory.ReadWrite.All`	Application and Delegated	Read and write directory data and manage users and groups
`Domain.Read.All`	Delegated	Read domains
`Group.Read.All`	Application and Delegated	Read all groups
`GroupMember.Read.All`	Application and Delegated	Read group memberships
`Group.ReadWrite.All`	Application and Delegated	Read and write all groups that the signed-in user is a member of, which includes creating, updating, and deleting groups.
`RoleManagement.Read.All`	Delegated	Read role management data for all RBAC providers
`RoleManagement.Read.CloudPC`	Delegated	Read Cloud PC RBAC settings
`RoleManagement.Read.Directory`	Application and Delegated	Read directory RBAC settings
`RoleManagement.ReadWrite.Directory`	Application and Delegated	Read and write directory role information. This includes creating, updating, and deleting directory roles and their assignments.
`User.Read`	Delegated	Sign in and read the user profile
`User.Read.All`	Application and Delegated	Read all user profiles
`User.ReadBasic.All`	Delegated	Read all users' basic profiles
`User.ReadWrite.All`	Application and Delegated	Read and update all users' profiles
`Organization.Read.All`	Application and Delegated	Read information about an organization in Microsoft 365.
`Organization.ReadWrite.All`	Application and Delegated	Read and change all settings and information for your Microsoft 365 organization, including directory and organizational data.

2 Verify that you can browse your Azure domain with the graph explorer using the account from Step 1. For more information, see the page on Microsoft Graph .

OpenText Identity Governance uses the Azure AD MS Graph collector to collect information from the SharePoint Team site. When you create a SharePoint Team site, a Microsoft 365 group is automatically created and changes to the SharePoint Team site, such as adding or removing users, are reflected in the associated Microsoft 365 group, and vice versa. These details are saved in Azure as a group. During data collection, OpenText Identity Governance collects information from Azure as part of a group and for each data collection, OpenText Identity Governance collects the SharePoint Team site information as part of the broader group collection.

NOTE: Only the SharePoint Team site is supported. OpenText Identity Governance does not support SharePoint Communication site.

4.2 Azure AD MS graph fulfillment

OpenText Identity Governance uses the Azure AD MS Graph fulfiller to automatically assign or remove permissions from user accounts and add or remove members from Microsoft 365 and Security groups. OpenText Identity Governance does not support adding or removing members from the Distribution List and Mail-enabled Security type of groups because Mail-enabled and distribution groups cannot be managed by Microsoft Graph group APIs.

The template supports the following fulfillment change requests:

- `ADD_APPLICATION_TO_USER`
- `ADD_PERMISSION_TO_USER`
- `REMOVE_ACCOUNT_PERMISSION`
- `REMOVE_PERMISSION_ASSIGNMENT`
- `REMOVE_ACCOUNT`
- `REMOVE_APPLICATION_FROM_USER`
- `REMOVE_ACCOUNT_ASSIGNMENT`

To fulfill these change requests, you need the following API permissions, ranging from least to the most privileged.

Change request	Permission class	Permission type	Permissions
Add application to user		Application and Delegated	◆ `User.ReadWrite.All` ◆ `Directory.ReadWrite.All`

Change request	Permission class	Permission type	Permissions
◆ Add permission to user ◆ Remove permission assignment ◆ Remove account permission	Directory Roles	Application and Delegated	◆ `RoleManagement.ReadWrite.Directory`
	Groups	Application and Delegated	◆ `GroupMember.ReadWrite.All`
	Service plan	Application and Delegated	◆ `LicenseAssignment.ReadWrite.All` ◆ `Directory.ReadWrite.All` ◆ `User.ReadWrite.All`
◆ Remove application from user ◆ Remove account ◆ Remove account permission		Delegated	◆ `User.ReadWrite` ◆ `User.ManageIdentities.All` ◆ `User.EnableDisableAccount.All` ◆ `User.ReadWrite.All` ◆ `Directory.ReadWrite.All`
		Application	◆ `User.ManageIdentities.All` ◆ `User.EnableDisableAccount.All` ◆ `User.ReadWrite.All` ◆ `Directory.ReadWrite.All`

The Azure MS Graph fulfiller has default mapping for some mandatory attributes. The Azure application requires these mandatory attributes to create an account. For the fulfillment to process successfully, you must add these mandatory attributes to the Fulfillment Context attribute. The following table provides the list of attributes.

Fulfillment Context Attributes	Attributes
Recipient	◆ User ID from Source ◆ Last Name ◆ First Name ◆ Full Name ◆ Email ◆ Employee Status
Account	◆ Account ID from Source ◆ Account Disabled
Permission	◆ Permission Type ◆ Permission ID from Source

NOTE: We recommend that while adding users to the Azure application, you provide a unique `mailNickName` for each user. The purpose of this is to prevent the error that can occur when you try to add users with the same first and last name. The ECMA script includes the logic for creating the unique `mailNickName`, but you can customize it to meet your requirements.

In addition to this list of attributes, you can configure other attributes in the collector template such as department, title, job codes, or workforce ID to match the requirements of your application. However, you must add them to the Fulfillment Context attribute. In addition, while configuring the fulfiller, go to **Fulfillment item configuration and mapping**, click **{..}**, then edit the transform script for **User Profile.**

In the transform script, you must add the native application key as `outUserProfile` and add the corresponding fulfillment context attribute key in the `outUserProfile` value. For example, for the attribute Workforce ID, edit the transform script to:

```
if(inUserProfile.workforceId) outUserProfile["employeeId"] =
inUserProfile.workforceId
```

NOTE: If you want to specify Workforce ID as the attribute for matching identities to accounts and permissions, then while configuring the collector template you must map Workforce ID to the native ID value, for example, `employeeId`, and set it as the matching rule.

OpenText Identity Governance uses the Azure AD MS Graph fulfiller to provision and deprovision users as a group from the SharePoint Team site. The following change requests are supported when provisioning and deprovisioning users as a group from the SharePoint Team site:

- `ADD_PERMISSION_TO_USER`
- `REMOVE_ACCOUNT_PERMISSION`
- `REMOVE_ PERMISSION_ASSIGNMENT`

5 CSV collectors and fulfiller

This chapter focuses on specific configuration-related information regarding the following collection and fulfillment templates:

- CSV Identity
- CSV Account
- CSV Permission
- CSV Fulfillment

For information about configuring CSV templates, see the following sections:

- Section 5.1, “Collecting from CSV files using Cloud Bridge,” on page 25
- Section 5.2, “CSV collectors,” on page 26
- Section 5.3, “CSV fulfillment,” on page 26

5.1 Collecting from CSV files using Cloud Bridge

NOTE: OpenText Identity Governance supports Cloud Bridge only in OpenText Identity Governance as a Service deployments.

Note that if you plan to collect from CSV files using Cloud Bridge, *you must place the files in the `conf` directory*. Navigate to the location where the Cloud Bridge Agent is installed, then place the CSV file in the `conf` directory or create different subfolders to separate the applications and place the file in the subfolders. For example:

- `/opt/netiq/mycba/conf/users.csv` or
- `/opt/netiq/mycba/conf/csv/oracle/users.csv`

Alternately, drop the CSV files to a subfolder of the `conf` directory that a network administrator previously configured as an NFS mount. This method enables data administrators from various locations, to place the CSV files without requiring access to the server where the Cloud Bridge Agent is installed.

IMPORTANT: We do not support changes within the Cloud Bridge container.

Before collecting, administrators need not restart the Cloud Bridge Agent when creating new subfolders in the `conf` directory or when placing new CSV files in those directories.

5.2 CSV collectors

A CSV file provides a simple method for storing user account or permissions information that cannot be collected from other data sources. You can include group, account, permission, or user data in the file. For all CSV collections, data administrators need to generate the CSV and make it available to the OpenText Identity Governance service through a file share, http, or local file system. To collect from a CSV file, you must specify the full path to the file.

When configuring the CSV collector:

- Start the root path file on the Services Parameters section of the template page with `/conf` and end with `/`. For example, `/conf/`.
- If you placed the CSV file in subfolders, append the subfolder names to the root file path. For example, `/conf/csv/oracle/`.
- Enter the CSV file name in the Collect Views section of the template page. For example, `users.csv` or `http://`*`ipaddress`*`/users.csv`

If you use a CSV file as an identity source, you might also want to instruct OpenText Identity Governance to map the collected users to their collected group memberships. The **Group Members (Users and Groups)** setting allows you to specify an attribute in the CSV file that you want to use for mapping users and groups to groups. However, you can use this setting only when a given value for the specified attribute is not used to identify both a user and a group. For example, if you export data from Active Directory to the CSV file, you can use DN as the Group Members attribute. Otherwise, you can use **Collect Group to User Membership** or **Collect Parent Group to Child Group Relationships** to map users or groups to groups. These two settings match the specified attribute in the collected user or group data, respectively.

In preparing a CSV file, ensure that any values written into a column of the file do not contain any carriage returns and line feeds, since these characters define record boundaries in the CSV file.

NOTE: The CSV collector supports TSV files. In the **Column Delimiter** field, enter the word `tab` in uppercase, lowercase, or any combination thereof. To collect from a CSV file, you must specify the full path to the file. Collection is not supported when the CSV collector is accessed through HTTPS connection.

5.3 CSV fulfillment

This fulfillment target creates a CSV file in the specified directory that contains the attributes you configured in the fulfillment target.

6 Google Apps collectors

This chapter focuses on specific configuration-related information regarding the following collection and fulfillment templates:

- Google Apps User Identity
- Google Apps User Account
- Google Apps User Permission

Google Apps manages users, groups, and organizational units, including assigned roles and privileges. Collecting identities from Google Apps is similar to other data sources. However, to collect permissions, OpenText Identity Governance pulls information from Google Groups, which resembles discussion-based groups similar to those available in Usenet.

To gather information about actual user groups, OpenText Identity Governance collects from the Organizations (organizational units) in Google Apps. These organizational units can contain nested units. The top-level organization is always called ‘root.’ During collection, OpenText Identity Governance translates the organizational units into OpenText Identity Governance-style groups. In OpenText Identity Governance, the root group lists all the users in that organizational unit. If you select one of the nested groups under the root group, OpenText Identity Governance lists only the individuals assigned to that group.

7 ILM collectors and fulfillers

ILM collectors and ILM fulfiller collect and fulfill identities (Users, Groups), accounts (Users), and permission (Groups) from OpenText™ Core Identity Lifecycle Manager SCIM endpoints. This chapter focuses on specific configuration-related information regarding the following collection and fulfillment templates:

- ILM Identity
- ILM Account
- ILM Permission
- ILM Fulfillment

For information about configuring ILM templates, see the following sections:

- Section 7.1, "ILM authentication methods and ordinals," on page 29
- Section 7.2, "ILM collectors," on page 30
- Section 7.3, "ILM fulfillment," on page 30

7.1 ILM authentication methods and ordinals

The ILM Collector and Fulfiller supports only the Generate Bearer/Access Token method for authentication. It is compatible with both the Client Credentials and Password Grant flows. For the current phase of OpenText Identity Governance and OpenText Core Identity Lifecycle Manager, we recommend password flow as the authentication method.

Roles and access: As part of the Password Grant flow, you must create user credentials in OpenText Core Identity Lifecycle Manager with the default role and sync it to OpenText Advanced Authentication. The user name and password along with the client ID and client secret are required for the Password Grant flow.

Client ID and client secret: Log in to your OpenText Advanced Authentication tenant. Within the tenant, open **ILMOauthApp** and retrieve the **Client ID** and **Client Secret** from the application. Additionally, copy the user name and password of the user you created in OpenText Core Identity Lifecycle Manager and synced to OpenText Advanced Authentication. You will need to enter them when configuring the template for password (resource owner credential) flow.

Other service parameters: Make sure your base URL matches with the validation URL of OpenText Core Identity Lifecycle Manager. When selecting the credential flow, use the `oauth2_aud_base` value as your **Resource** to specify the audience for which the authentication token is being requested. It indicates the API or service to which the token is intended to grant access. The audience typically refers to the unique identifier (such as the URI) of the resource or API you are trying to access.

When using Cloud Bridge, you must also specify a unique ordinal for each authentication method. Use the following table to understand the ordinal number that you need to specify for the **Generate Bearer Token** field.

The following table lists the available authentication types and related credentials:

Ordinal (Credential Position)	Authentication Type	Credential Set
5	Bearer Token	◆ User Name ◆ Password
6	Bearer Token	◆ Client ID ◆ Client Secret

7.2 ILM collectors

The ILM identity, account, and permission collectors use unique authentication methods. In addition to specifying the authentication method, you might need to change attribute mapping when configuring the collector template. Like SCIM collectors, ILM collectors also support singular attributes, complex singular attributes, complex multi-valued attributes, and extensions. If your application supports any other attributes or extensions different from those mentioned in the SCIM protocol, you can change the attribute mapping in the template by using delimiters. You can use ':' (colon) for attributes, for example, `emails:work:`*`value`*, and '+' (plus) for extensions, for example, `urn:ietf:params:scim:schemas:extension:enterprise:2.0:User+`*`department`*.

To successfully map accounts and permissions to identities, you may specify any unique account attribute as the mapping attribute. For example, `userName, emails:primary:value`. Because ILM collectors also use SCIM protocols, they record in batches of up to 999 records, and the default batch collection session timeout value is set to 60 seconds.

7.3 ILM fulfillment

The ILM fulfiller has default attribute mappings that help you fulfill different types of change requests. In addition to change request types supported by the SCIM fulfiller, the ILM fulfiller also supports the `USER_PROFILE_MODIFICATION` change request type. All required information for this change request is provided in the **User profile modification payload** field.

In addition to configuring the template and editing transformation scripts as needed, you must add attributes in the Fulfillment Context Attributes area to successfully fulfill change requests. For more information, see the following sections:

- Section 7.3.1, "ILM transformation scripts and payloads," on page 31
- Section 7.3.2, "Mandatory fulfillment context attributes," on page 32

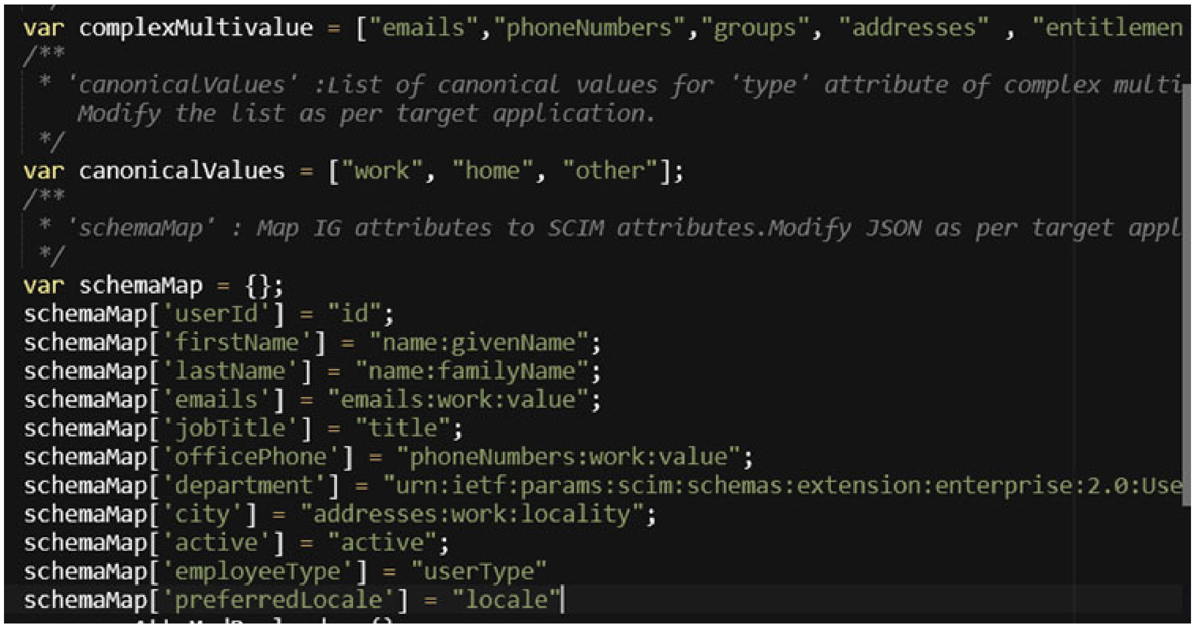
7.3.1 ILM transformation scripts and payloads

The ILM fulfiller template allows you to edit the transformation scripts to build the required payload for the change requests for generic fulfillment, user profiles, permissions, and accounts. The ECMA script includes comments that guide you through the payload generation process. After you generate the payload, OpenText Identity Governance sends the payload for fulfillment. The ILM fulfiller template includes the following payloads for the specified change requests:

Payload	Change Requests
User generation payload	`ADD APPLICATION USER`
User profile modification payload	`MODIFY USER PROFILE`
Permission generation payload	◆ `ADD PERMISSION TO USER` ◆ `REMOVE ACCOUNT PERMISSION` ◆ `REMOVE USER PERMISSION`
Account generation payload	`REMOVE ACCOUNT`

Click **{...}** next to the value of a parameter to define and edit data transformations scripts. Expand the Transformation Scripts view to see a list of included scripts. The following figure provides an example of the ILM transformation script.

Figure 7-1 *ILM user_modification_payload script example*



```
var complexMultivalue = ["emails","phoneNumbers","groups", "addresses" , "entitlemen
/**
 * 'canonicalValues' :List of canonical values for 'type' attribute of complex multi
   Modify the list as per target application.
 */
var canonicalValues = ["work", "home", "other"];
/**
 * 'schemaMap' : Map IG attributes to SCIM attributes.Modify JSON as per target appl
 */
var schemaMap = {};
schemaMap['userId'] = "id";
schemaMap['firstName'] = "name:givenName";
schemaMap['lastName'] = "name:familyName";
schemaMap['emails'] = "emails:work:value";
schemaMap['jobTitle'] = "title";
schemaMap['officePhone'] = "phoneNumbers:work:value";
schemaMap['department'] = "urn:ietf:params:scim:schemas:extension:enterprise:2.0:Use
schemaMap['city'] = "addresses:work:locality";
schemaMap['active'] = "active";
schemaMap['employeeType'] = "userType"
schemaMap['preferredLocale'] = "locale"
```

As the example shows, the ECMA script includes comments that guide you through the payload generation process and three input categories:

- **complexMultivalue:** List containing all SCIM complex multi-valued attributes.
- **CanonicalValues:** List of canonical values for the 'type' attribute of complex multi-valued attributes.

Example:`{ "emails" : [ { "value" : "johndoe@opentext.com", "type" : "work" }, { "value : "johndoe123@gmail.com", "type" : "home" },{ "value : "jdoe@gmail.com", "type" : "other" }]}`

In this example, canonical values are `work, home, other`.

- **Schema map attribute:** This attribute maps the OpenText Identity Governance attribute to the SCIM attribute namespace. By default, required attributes are added. You can extend the default schema and add custom attributes or remove attributes. For more information about extending the schema, see "Extending the OpenText Identity Governance Schema (https://www.microfocus.com/documentation/identity-governance-and-administration/igaas/user-guide/b1ljwwfe.html)" in the *OpenText Identity Governance as a Service User and Administration Guide (https://www.microfocus.com/documentation/identity-governance-and-administration/igaas/user-guide/front.html)*.

When using custom attributes:

- Enable **Allow to be reviewed** when adding the custom attribute.
- Select **Configuration > Fulfillment Context Attributes**, then add the custom attribute.
- Add the custom attribute to the user_modification_payload script by:
  - Adding the custom attribute to schemaMap
  - Mapping the custom attribute to the SCIM namespace
  - (Conditional) If the custom attribute is complex multi-valued, adding the custom attribute to the complexMultivalue list, and also adding its canonical values to the canonicalValues list

7.3.2 Mandatory fulfillment context attributes

For the ILM fulfillment to process successfully, you must add certain mandatory attributes to the Fulfillment Context attribute area. The following table provides the list of required attributes.

Fulfillment Context Attributes	**Attributes**
User (Requester and Recipient)	◆ Active ◆ City ◆ Department ◆ Email ◆ Employee Type ◆ First Name ◆ Last Name ◆ Phone - Office ◆ Preferred Locale ◆ Title ◆ User ID from Source ◆ Workforce ID
Account	◆ Account ID from Source ◆ Account Name

Fulfillment Context Attributes	Attributes
Permission	◆ Permission ID from Source ◆ Permission Name

8 OpenText Identity Manager collectors and fulfillers

This chapter focuses on specific configuration-related information regarding the following collection and fulfillment templates:

- Identity Manager Identity
- Identity Manager AE Permission
- Identity Manager Automated Fulfillment
- Identity Manager Workflow
- IDM Entitlement Account
- IDM Entitlement Permission
- IDM Entitlement Fulfillment
- Identity Manager Dxcmd Fulfillment for Active Directory

For information about configuring the OpenText Identity Manager templates, see the following sections:

8.1 Authentication methods and ordinals for IDM AE permission collectors and IDM automated fulfillment targets

The Identity Manager AE Permission collector requires both LDAP and user application credentials. All objects including roles collected using this collector are represented as permissions in the OpenText Identity Governance catalog. Note that the Identity Manager Automated Fulfillment fulfiller also uses the same credentials.

Log in to the Cloud Bridge URL, then specify the ordinal when adding credentials.

Use the following table to understand the order and ordinal number that you need to specify for this collector.

Ordinal (Credential Position)	Authentication type	Credential Set
0	LDAP	◆ User Name used to connect to Identity Vault Server (cn=admin,ou=sa,o=system) ◆ Password
1	User Application	◆ User Name used to connect to User Application (cn=uaadmin,ou=sa,o=data) ◆ Password

8.2 Identity Manager AE permission collectors

The Identity Manager AE Permission collector is an Application Source collector that creates the base OpenText Identity Managerapplication in OpenText Identity Governance and automatically generates subordinate applications that represent IDM Drivers, such as the CloudAD Driver and SAP User Management Driver, that support Identity Manager entitlements.

IMPORTANT: No other application source permission collector provides *automatic generation* of subordinate applications or accounts. This collector uses both LDAP calls into eDirectory and SOAP calls to the user applications to collect data. Due to the complexity of the relationships managed by this collector, proceed with caution when changing the default values and mappings.

In SaaS environments, when using the Cloud Bridge to collect data from your on-premises data centers, you will need to specify ordinals for the respective authentication method in the Cloud Bridge user interface (http:*//localhost (CBA IP address or DNS name)*:8080).

Before collecting Identity Manager AE permissions, ensure that you have installed the OpenText Identity Manager applications. Additionally, when using AD Driver with Identity Manager AE, ensure that the Remote Loader is running.

When configuring service parameters, ensure that you include the port number that you use to connect to your OpenText Identity Manager system in the **User Application Base Provisioning Service URL** field. Enter comma-separated values in the **Additional permission attributes to collect** field when you want to collect multiple attributes from Roles, Resources, Groups, and Container-type permissions in addition to the default attributes. When adding these additional permission attributes, you must also include the attributes in the collector views.

When the Identity Manager AE Permission collector collects any User record from the OpenText Identity Manager application that has an association with a subordinate application (through the DirXML Association attribute on the User), it receives an Account assignment for that subordinate application. The Identity Manager AE Permission collector also automatically maps the User record to the OpenText Identity Manager User.

If, after testing the connection and collecting data, you do not see the expected data in the OpenText Identity Governance Catalog, verify that your **Account Collect LDAP Search Filter** is configured correctly in the template, then use LDAP search from the command line or LDAP browser to confirm that the missing data is still available in your data source. You can also directly call the SOAP endpoint to get the refreshed values of the Identity Manager AE system attributes that are used for mapping.

NOTE: NOTE: Depending on the certificates and how the HTTPS connections are defined for the Tomcat servers utilized by the Identity Applications and OpenText Identity Governance, you might see the following error when testing connection, testing collection, or collecting permissions using the Identity Manager AE Permission Collector:

```
javax.net.ssl.SSLHandshakeException
```

If this error occurs, *add* additional ciphers to the list of accepted ciphers, such as `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`, in the HTTPS connection definition for the Tomcat server used by the Identity Applications. If needed, use HTTPS debugging, LAN trances, or other debugging methods to determine which cipher to add.

8.3 Identity Manager automated fulfillment

The Identity Manager Automated Fulfillment template enables automatic fulfillment of change requests related to Identity Manager AE permissions. Specify whether you want to use automated provisioning with manual fulfillment or a workflow as the fallback method, then specify the values associated with the fallback method. For more information, see “Automatically Fulfilling the Changeset (https://www.microfocus.com/documentation/identity-governance-and-administration/igaas/user-guide/fulfill-changesets.html#fulfill-changeset-automatic)” in the *OpenText Identity Governance as a Service User and Administration Guide (https://www.microfocus.com/documentation/identity-governance-and-administration/igaas/user-guide/front.html)* .

NOTE: When an Identity Manager AE permission is requested in the OpenText Identity Governance Access Request, the `idmDn` attribute of the requesting user is utilized in the RBPM SOAP request as the `<`*`ser:requester`*`>`. If this value is *not* a valid user DN in the target OpenText Identity Manager system, the fulfillment request will fail.

8.4 Identity Manager Entitlement collectors

The Identity Manager Entitlement collectors are for users who want to use OpenText Identity Governance to provision or revoke accounts and permissions. The entitlement collectors collect accounts and permissions from OpenText Identity Manager using IDM drivers that support entitlements such as Azure, Workday, and SCIM drivers. Like the other OpenText Identity Governance collectors, the entitlement collectors map accounts and permissions to identities by association or other attributes. To successfully collect accounts and permissions:

- You must have collected identities from OpenText Identity Manager, and
- All supported drivers must be running

NOTE: For list of supported drivers and packages, see Appendix A, “Supported OpenText Identity Manager drivers and packages,” on page 89.

You can use the account collectors to collect accounts and the permission collectors to collect permissions and their assignments. For example, the permission collector can collect the building access permission (list of buildings) and assignments (who can access the building).

In addition to IDM credentials, the LDAP distinguished name of the entitlement in the IDM driver is also a mandatory field for the entitlement collectors.

Typically, attribute mappings are preconfigured and have built-in fallback options. For example, by default, the collector maps the account or permission name to the IDM display name. If the collection process does not find the display name, the collector automatically maps the account or permission name to other available attributes such as the description. However, you do need to change the default **Account-User Mapping** value GUID to **Object GUID** and the default **Permission-Account or User Mapping** value association to **Account ID from source.**

8.5 IDM Entitlement fulfillment

If you are using the IDM Entitlement Fulfillment target with IDM Advanced Edition, you must set the **Entitlement Result Purge type** to Previous in the GCV values of the User Application Driver. For more information, see, Global Configuration Values (https://www.netiq.com/documentation/identity-console/identity_console-admin/data/t4esr4m9uxod.html#t4esrke4pl5b) in the *Identity Console Administration Guide*. The target supports only the following fulfillment change requests:

- `ADD_APPLICATION_TO_USER`
- `ADD_PERMISSION_TO_USER`
- `REMOVE_ACCOUNT_PERMISSION`
- `REMOVE_PERMISSION_ASSIGNMENT`
- `REMOVE_ACCOUNT`

When a change request is sent to OpenText Identity Manager for fulfillment, the fulfiller modifies the User Attribute `DirXML-EntitlementRef.` The IDM engine then sends an event to the driver to ensure that the entitlement is fulfilled.

To successfully fulfill entitlement-related change requests:

- Identities must have been collected from OpenText Identity Manager
- Users must still be present in Identity Manager
- All the fulfillment context attributes required for Recipient (User), Account, and Permission profiles must be specified

Occasionally, when you remove an account from a database, even though fulfillment is successful, OpenText Identity Governance might display the status as Not Fulfilled, Verification Error. This occurs because the value returned by the database might not be consistent with the values the JDBC driver expects. To avoid this issue, ensure that the account status in the entitlement configuration for the driver displays the following values:

- For MS SQL and Oracle: `<account-status active="0" inactive="1" source="read-attr" source-name="Login Disabled"/>`
- For PostgreSQL: `<account-status active="FALSE" inactive="TRUE" source="read-attr" source-name="Login Disabled"/>`

8.6 Identity Manager Dxcmd fulfillment

The configuration for the Dxcmd fulfiller template is similar to any other template, but you must ensure the Active Directory driver is installed in the same OpenText Identity Manager application where you are sending the change requests for fulfillment.

When configuring the AD Account Collector and Permission Collector, set the service parameters as you would for any other collector. In the Collect Account view, ensure the **Privileged Account** is set to `false`. In the Collect Permission view, the **Permission-Account or User Mapping** is mapped to `member`. Additionally, map the **Parent Permission ID** to `memberOf` and the **Target attribute used for provisioning** to `member`.

9 JDBC collectors and fulfillers

JDBC enables applications to connect to different types of databases. This chapter focuses on specific configuration-related information regarding the following collection and fulfillment templates:

- JDBC Identity
- JDBC Account
- JDBC Permission

  NOTE: OpenText Identity Governance provides the following generic and database-specific identity, account, and permission templates:

  - DB2
  - Generic
  - MySQL
  - Non-specific
  - Oracle
  - PostgreSQL
  - SQLServer
  - Sybase

- JDBC Generic DB Fulfillment
- JDBC Oracle Fulfillment
- JDBC PostgreSQL Fulfillment
- JDBC SQL Server Fulfillment

Configuration fields and values of the collector and fulfiller templates will differ based on the type of database template you select. Use the default values and tooltips to configure the templates as needed. For information about unique settings or procedures when configuring JDBC templates, see the following sections:

- Section 9.1, "Requirements when using Cloud Bridge," on page 41
- Section 9.2, "JDBC collectors," on page 42
- Section 9.3, "JDBC fulfillment," on page 42

9.1 Requirements when using Cloud Bridge

NOTE: OpenText Identity Governance supports Cloud Bridge only in OpenText Identity Governance as a Service deployments.

If you plan to collect data from a JDBC source using Cloud Bridge, *you must save the required third-party connector libraries to the bridgelib folder relative to the Cloud Bridge Agent*.

1. Ensure that the JAR files are placed in the `/opt/netiq/bridgelib` folder.
2. (Conditional) If the file is not already in the bridgelib folder, place the `dist-collectors.jar` in the same bridgelib folder.

   NOTE: If you do not have the `dist-collectors.jar`, please contact the SaaS Support team.

3. Restart the Cloud Bridge Agent.

9.2 JDBC collectors

To collect data from a JDBC source, OpenText Identity Governance needs the appropriate third-party connector libraries to be installed on the OpenText Identity Governance server.

When using the non-specific collector template and specifying database type as Microsoft SQL Server or Oracle, change the default value and set **Use SSL Secure Connection?** to **Yes** *only* when the respective database is configured for secure SSL connection. For direct connections to these database servers, do not change the default setting.

9.3 JDBC fulfillment

OpenText Identity Governance uses the JDBC, Oracle, SQL Server, and PostgreSQL fulfillment templates to automatically fulfill change requests. OpenText Identity Governance uses the generic fulfillment template for all other databases, such as MySQL or SyBase. The appropriate third-party connector libraries must be installed on the OpenText Identity Governance server before you can use the JDBC generic fulfillment template. The generic template allows you to edit the transform script that builds the required payload to successfully process change requests.

The JDBC fulfillment template supports all change requests. The JDBC fulfillment is certified with the following database versions:

JDBC fulfillment type	Supported version
JDBC Oracle	Oracle 21c
JDBC PostgreSQL	PostgreSQL 16
JDBC SQL Server	MS SQL 2022
JDBC Generic DB	MySQL 8.0.x

10 Microsoft Teams collector and fulfiller

This chapter focuses on specific configuration-related information regarding the following collection and fulfillment templates:

- MS Teams Permission Collector
- MS Teams Fulfillment

For information about configuring MS Teams templates, see the following sections:

- Section 10.1, "Microsoft Teams collectors," on page 43
- Section 10.2, "Microsoft Teams fulfillment," on page 45

10.1 Microsoft Teams collectors

The Microsoft Teams application is a subordinate application and uses the Azure Active Directory database. It consists of teams and channels with members of their own. MS Teams further divides members into team and channel members, or team and channel owners, with higher privileges. Teams are public and private and channels are standard and private. Each team can have a number of channels with one default standard channel.

While collecting data from the Microsoft Teams application, you must use the Azure AD MS Graph collector for collecting accounts and identities and use the MS Teams collector to collect teams, channels, their members, and the associated permissions. However, for the collector to work, you must have the following API permissions in Azure Active Directory.

Resource	Type	Permission	Description
Team	Application	`TeamSettings.Read.Group`	Read team's settings
	Application	`TeamSettings.ReadWrite.Group`	Read and write team's settings
	Application	`User.Read.All`	Read all user profiles
	Application	`User.ReadWrite.All`	Read and write all user profiles
	Application and Delegated	`Team.ReadBasic.All`	Read names and descriptions of all teams
	Application and Delegated	`TeamSettings.Read.All`	Read all teams settings
	Application and Delegated	`TeamSettings.ReadWrite.All`	Read and change all teams settings
	Application and Delegated	`Group.Read.All`	Read all groups
	Application and Delegated	`Group.ReadWrite.All`	Read and write all groups

Resource	Type	Permission	Description
	Application and Delegated	`GroupMember.ReadWrite.All`	Read and modify the membership of groups within an organization. Also, manage group memberships, add and remove members.
	Application and Delegated	`Directory.Read.All`	Read all directory data
	Application and Delegated	`Directory.ReadWrite.All`	Read and write directory data
	Application	`Directory.AccessAsUser.All`	Access the directory as the signed-in user
	Application	`TeamMember.Read.Group`	Read team's members
	Application and Delegated	`TeamMember.Read.All`	Read all team members
	Application and Delegated	`TeamMember.ReadWrite.All`	Add, remove, and change roles for members of all teams
	Application	`TeamMember.ReadWriteNonOwnerRole.All`	Add and remove members with non-owner roles for all teams
Channel	Application	`ChannelSettings.Read.Group`	Read channel data of a team
	Application	`ChannelSettings.ReadWrite.Group`	Update channel data of a team
	Application and Delegated	`Channel.ReadBasic.All`	Read all channel names and descriptions
	Application and Delegated	`ChannelSettings.Read.All`	Read all channel data of a team
	Application and Delegated	`ChannelSettings.ReadWrite.All`	Read and write all channel data
	Application and Delegated	`Group.Read.All`	Read all groups
	Application and Delegated	`Group.ReadWrite.All`	Read and write all groups
	Application and Delegated	`Directory.Read.All`	Read directory data
	Application and Delegated	`Directory.ReadWrite.All`	Read and write directory data
	Application and Delegated	`ChannelMember.Read.All`	Read channel members
	Application and Delegated	`ChannelMember.ReadWrite.All`	Add, remove, and change roles for members of all channels

IMPORTANT: The Microsoft Teams collector does not collect data for itself. So, you *must* enable the Azure Active Directory data source to collect permissions from MS Teams.

You have the option to configure the MS Teams collector as a hierarchical structure and map the attribute **Unique Application ID** with the `applicationId`. Ensure that the `outputValue` in the ECMA script is mapped to the name of the collector. For example, `outputValue='MS_Teams'`. Also, configure the MS Teams Permission collector template mandatory attribute mappings, such as `ID`, and `objectType`. ID is the unique ID from a team or a channel, and objectType indicates whether the object is for teams or channels.

Occasionally, while collecting data using the MS Teams collector, the collection might fail with an error message. This occurs because of issues such as an application timeout when the response from the Microsoft Teams API takes a long time to return or a backend error when the Microsoft Teams API is not able to process the request. Check your configuration, change the timeout value, view logs and audit events, and try again.

10.2 Microsoft Teams fulfillment

If you have the appropriate permissions in Azure Active Directory, you can fulfill the following change requests:

- `ADD PERMISSION TO USER`
- `REMOVE ACCOUNT PERMISSION`
- `REMOVE PERMISSION ASSIGNMENT`

You can add or remove a member only from a private channel. However, before adding a member to a channel, ensure that the member is already a part of the team. When you add a user to a team, the Microsoft Teams fulfiller adds the user automatically to all standard channels under the team, as a member.

NOTE: To avoid unexpected behavior from the application, we recommend that you do not add a team and a channel member in the same request.

To fulfill these change requests, you need the following API permissions.

Resource	**Type**	**Permission**	**Description**
Teams	Delegated and Application	`TeamMember.ReadWrite.All`	Read and write data for all team members in the organization.
Channel	Delegated and Application	`ChannelMember.ReadWrite.All`	Read and write data related to channel members across the organization.

You can assign the user the role of an owner. To do so, you need to customize the request form and add ‘owner’ as **Data Source Values** and ‘roles’ as **Label**, then publish the form. This will allow you to select the role as ‘owner’ when you request permission for the user. For information about customizing forms using Form Builder, see Creating a Request or Approval Form (https://

www.netiq.com/documentation/iga-form-builder/iga_form_builder/data/create-forms.html) in the *Administrator's Guide to Form Builder (https://www.netiq.com/documentation/iga-form-builder/iga_form_builder/data/bookinfo.html)*. Additionally, while configuring **Fulfillment item configuration and mapping** in the template, you must add `"flowdata"` for the attribute **Permission Profile**. For example, add `["flowdata", "permissionProfile"]`.

NOTE: To assign a user as an owner you need to create custom forms for each team and channel separately.

For the fulfillment to process successfully, you must add the following attributes to the fulfillment context attribute:

Fulfillment Context Attributes	Attributes
Recipient	◆ User ID from Source ◆ Full Name ◆ Employee Status ◆ Last Name ◆ First Name ◆ Email
Account	◆ Account ID from Source ◆ Account Disabled
Permission	◆ Permission Type ◆ Permission ID from Source ◆ Permission Name

11 REST GitHub collectors and fulfiller

This chapter focuses on specific configuration-related information regarding the following collection and fulfillment templates:

- REST GitHub Account
- REST GitHub Organization Permission
- REST GitHub Repository Permission
- REST GitHub Team Permission
- REST GitHub Fulfillment

For additional information about configuring REST GitHub and Generic templates, see the following sections:

- Section 11.1, “Required minimum rights for integration with GitHub,” on page 47
- Section 11.2, “REST GitHub authentication methods,” on page 47
- Section 11.3, “REST GitHub collectors,” on page 48
- Section 11.4, “REST GitHub fulfillment,” on page 49

11.1 Required minimum rights for integration with GitHub

To access all GitHub endpoints, for the authentication types, you must use the credentials of a GitHub Enterprise Administrator, also known as the Site Administrator. The GitHub collector can collect all organization.

11.2 REST GitHub authentication methods

NOTE: OpenText Identity Governance supports Cloud Bridge only in OpenText Identity Governance as a Service deployments.

If you use Cloud Bridge to collect data from your on-premises data centers, you must specify ordinals for the respective authentication method in the Cloud Bridge user interface (http:*//localhost (CBA IP address or DNS name)*:8080).

The GitHub account and permission collector supports two authentication types: Basic Auth and Access Token. The collector collects all organizations.

NOTE: You must have installed Cloud Bridge 1.10 to collect with the Rest connector 1.0.2.0000 version.

Log in to the Cloud Bridge URL, then specify the ordinal when adding credentials.

Ordinal (Credential Position)	Authentication Type	Credential Set
3	Basic Auth	◆ User Name ◆ Password
4	Access Token **NOTE:** When the access token expires, replace it with a new access token.	◆ Access Token Header ◆ Access Token

IMPORTANT: For the access token, the user provides the token to connect to the target application, whereas, for the bearer token, the connector generates the token. When the access token expires, replace it with a new access token.

11.3 REST GitHub collectors

NOTE: OpenText Identity Governance currently supports GitHub Enterprise on-premises edition, version 3.8.3.

Like the other OpenText Identity Governance collectors, the REST GitHub collectors map accounts and permissions to identities by association or other attributes. To successfully map accounts and permissions to identities, identities must be collected using the identity source that is configured in the GIT server. The REST GitHub permission collector collects all organizations, teams, repositories, the permission to permission association, and also the holder association. The collector has a batch size limit of 100 records.

The REST GitHub collector template includes mandatory attribute mappings suitable for the target application. However, you can configure other attributes and then edit the transform script to build the required payload. For example, for GitHub accounts, if you want to map 'email' for the **Account-User Mapping** attribute, you need to map **Account-User Mapping** with `user` and write the script as follows to parse the email from the user JSON:

```
var user = JSON.parse(inputValue) outputValue = user['email'];
```

Apart from the mapped attributes, you can add additional attributes for `organization`, `teams`, and `repository` by parsing values from the organization, teams, or repository JSONs.

If you use Cloud Bridge to collect data from your on-premises data centers, you must specify ordinals for the respective authentication method in the Cloud Bridge user interface (http:*//localhost (CBA IP address or DNS name)*:8080).

NOTE: You must be using OpenText Identity Governance 4.2 or Cloud Bridge 1.10 to collect using the latest version of REST GibHub collector.

11.4 REST GitHub fulfillment

OpenText Identity Governance uses the REST GitHub fulfiller to add or remove members from an organization, or a team, or add or remove a collaborator from a repository. When a user is added to an organization or a team the default role assigned is of a “member”, and for a repository, it is “read”. However, members can log in to the GitHub application and change their roles as needed.

Users can get access to a repository directly as collaborators, or when they are members of an organization or a team. As members, they automatically inherit the permission to access the organization and team repositories. So, when you want to remove a collaborator from a repository, or a member from a team, ensure that the repository permission is not inherited from an organization or a team. For the fulfillment verification to be successful, you must remove the member from the parent organization or team so the member loses the child permission, which means the repository permission.

NOTE: The term “collaborator” is specific to GitHub and it refers to a user who is given access to a repository directly. For more information, see the GitHub Docs .

The REST GitHub fulfiller supports the following change requests:

- `ADD PERMISSION TO USER`
- `REMOVE PERMISSION ASSIGNMENT`
- `REMOVE PERMISSION FROM ACCOUNT`

For the fulfillment to process successfully, you must add these mandatory attributes to the Fulfillment Context Attributes area. The following table provides the list of attributes.

Fulfillment Context Attributes	**Attributes**
Account	◆ Account ID from Source ◆ Account Disabled ◆ Account Aliases
Permission	◆ Permission ID from Source ◆ Permission Type ◆ Permission Name

12 REST OTDS collectors and fulfiller

This chapter focuses on specific configuration-related information regarding the following collection and fulfillment templates:

- REST OTDS Identity
- REST OTDS Account
- REST OTDS Hybrid Permission
- REST OTDS Permission
- REST OTDS Fulfillment

OpenText™ Directory Services functions as a central repository for user and group identity information. It enables synchronization and single sign-on capabilities across multiple OpenText applications, making task management simple and ensures security by automatically syncing with identity providers. For more information, see OpenText Directory Services (https://developer.opentext.com/ce/products/opentext-directory-services).

For information about configuring the OpenText Directory Services templates, see the following sections:

- Section 12.1, “Required minimum rights for integration with OpenText Directory Services,” on page 51
- Section 12.2, “OpenText Directory Services authentication method and ordinal,” on page 52
- Section 12.3, “Extending the schema for data synchronization,” on page 52
- Section 12.4, “Configuring the OpenText Directory Services identity collector,” on page 52
- Section 12.5, “Configuring the OpenText Directory Services application collector,” on page 53
- Section 12.6, “OpenText Directory Services fulfillment,” on page 55

12.1 Required minimum rights for integration with OpenText Directory Services

When you create an OAuth Client in OpenText Directory Services, it automatically allows you to collect roles, users, accounts, and groups. However, to collect access roles, you need to add the OAuth client to the Administration group.

12.2 OpenText Directory Services authentication method and ordinal

The OpenText Identity Governance OTDS collector supports the OAuth2 protocol for authentication and bearer token as its authentication type, with the grant type as Client Credential Flow. When using the bearer token authentication method, you must specify the client ID, client secret, and token type.

Log in to the Cloud Bridge URL, then specify the ordinal when adding credentials.

Use the following table to understand the ordinal number that you need to specify for this collector.

Ordinal (Credential Position)	Authentication Type	Credential Set
6	Bearer Token	◆ Client ID ◆ Client Secret

12.3 Extending the schema for data synchronization

For collecting data from OpenText Directory Services and mapping the data to the OpenText Identity Governance catalog, you must configure your LDAP system, for example, Active Directory, as a synchronized partition in OpenText Directory Services. In some scenarios you might need to extend the schema of your LDAP system to accommodate additional attributes. For example, if you want to synchronize xECM's two permissions `supplementalMarking` and `securityClearance` from OpenText Directory Services to xECM through your LDAP system, in this case Active Directory, then you need to extend the schema.

12.4 Configuring the OpenText Directory Services identity collector

When you configure the OTDS collector you must specify the token type in the service parameters to generate a token in OTDS.

Then, in the **Collect Identity** view:

- If you specify both the **User Partition Name** and the **DN of the Organizational Unit**, OpenText Directory Services prioritizes the Organizational Unit and OpenText Identity Governance collects users from it.
- If you specify only the user partition name, OpenText Identity Governance collects users from that partition.
- If you do not specify a value, OpenText Identity Governance collects users from all partitions in OTDS.

To collect additional attributes from OpenText Directory Services that are not included in the default collector template, specify the attributes in the **Additional User Profile Attribute** field. If a matching attribute exists in OpenText Identity Governance, map the additional user profile attribute to it. Otherwise, add a new Identity Governance attribute and map the additional user profile attribute to the new OpenText Identity Governance attribute.

12.5 Configuring the OpenText Directory Services application collector

The OpenText Directory Services application collector collects accounts, permissions, and hybrid permissions. The **REST OTDS Permission** template allows you to collect hierarchical permissions. While configuring the service parameters in the template, enable the option 'Populate Nested Permission Assignments'. However, when you enable this option, OpenText Identity Governance prevents you from fulfilling change requests to remove permissions from accounts and permission assignments. Additionally, the template allows you to select various permission types from the **Collect OTDS Permissions** view, including groups, roles, and access roles and in any combination.

12.5.1 Configuring the OpenText Directory Services hybrid permission collector

By default the hybrid permission collector template script is configured to collect permissions from xECM and synchronize it to OpenText Directory Services user attributes, supplemental markings, and security clearance. To collect these permissions you must generate a CSV file from xECM first and use the file as the input value. After collection, these permissions are mapped as user profile attributes in OpenText Directory Services. Any changes you make to these attributes in OpenText Directory Services are synchronized to xECM.

NOTE: Before configuring the REST OTDS Hybrid Permission template, you must first generate a CSV file with the security clearance and supplemental marking permission from xECM.

To generate the CSV file:

1. From any client application that is used for API testing, for example, Postman, send a ‘`Get`’ request for security clearance using the following request URL:

   `https://{{host}}/cs/cs/api/v1/securityclearances`

   Replace `{{host}}` with the xECM hostname.

   For more information on how to authenticate users and get security clearance and supplemental marking, refer to the following links:

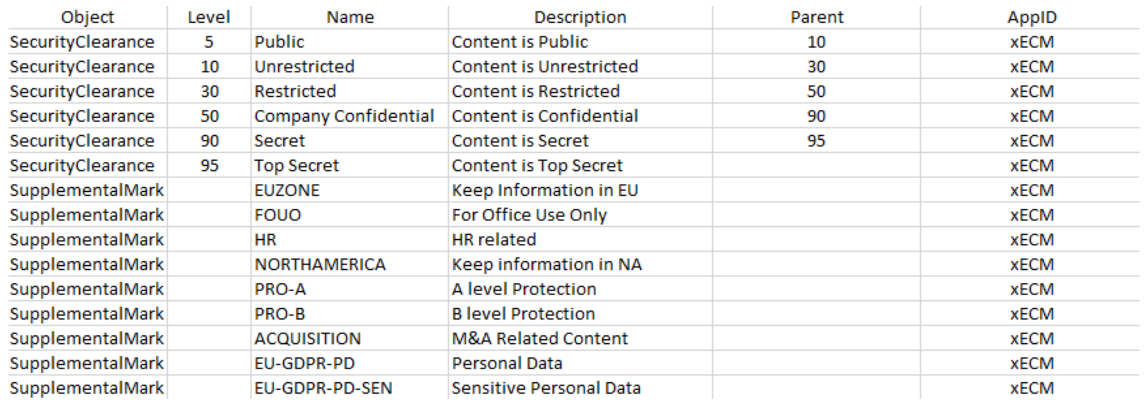
   Authenticate User (https://developer.opentext.com/ce/products/extended-ecm/apis/content-server-241-rest-api#auth/authenticate)

   Get Security clearance and supplemental marking (https://developer.opentext.com/ce/products/extended-ecm/apis/records-management-234-rest-api#SecurityClearance/getSecurityClearance)

2. When you have the data, copy the JSON result and convert it to a CSV file.
3. Review the CSV file output and make any necessary adjustments. This may involve aligning the column headings with those specified in the **Row Pattern** field for Hybrid Permission > **Collect Permission** view or consolidating relevant data under a single column.

   The following shows a sample CSV file after making changes:

Figure 12-1 *Sample CSV file*



Object	Level	Name	Description	Parent	AppID
SecurityClearance	5	Public	Content is Public	10	xECM
SecurityClearance	10	Unrestricted	Content is Unrestricted	30	xECM
SecurityClearance	30	Restricted	Content is Restricted	50	xECM
SecurityClearance	50	Company Confidential	Content is Confidential	90	xECM
SecurityClearance	90	Secret	Content is Secret	95	xECM
SecurityClearance	95	Top Secret	Content is Top Secret		xECM
SupplementalMark		EUZONE	Keep Information in EU		xECM
SupplementalMark		FOUO	For Office Use Only		xECM
SupplementalMark		HR	HR related		xECM
SupplementalMark		NORTHAMERICA	Keep information in NA		xECM
SupplementalMark		PRO-A	A level Protection		xECM
SupplementalMark		PRO-B	B level Protection		xECM
SupplementalMark		ACQUISITION	M&A Related Content		xECM
SupplementalMark		EU-GDPR-PD	Personal Data		xECM
SupplementalMark		EU-GDPR-PD-SEN	Sensitive Personal Data		xECM

4 Save the CSV file on your local drive, then copy it to a folder on the OpenText Identity Governance server.

5 Log in to OpenText Identity Governance, then navigate to **Data Sources** > **Applications**.

6 Create an application source or click the application name from the Application Sources page.

7 Select the **OTDS Hybrid** permission template.

8 Configure service parameters as required.

9 Expand the **Collect Permission** view and in the **Permission File Name** field specify the full path to the CSV file.

10 Specify details as necessary.

11 Expand the **Collect Holder to Permission Mapping** view and select **Collect this data?** to collect holders from OpenText Directory Services.

12 Specify details as necessary and save.

In the **Collect Holder to Permission Mapping** view, the **User Profile Permission Attribute** and the **Holder Permission(s)** fields should have corresponding attribute values. The default values in the template are `otSupplementalMarking` and `otSecurityClearanceLevel`. However, you can modify or change these values. For example, if you modify the **User Profile Permission Attribute** value from `oTSupplementalMarking` to `SupplementalMarking`, you must update the **Holder Permission(s)** field as well. Additionally, update the holder permission ECMA script and change all instances of `oTSupplementalMarking` to `SupplementalMarking` as given in this sample script.

```
/* start of generated script to extract variables from inputValue */

/* NOTE: any code inserted within this block will get overridden if inputs
are regenerated */

var vals={};

// inputValue is string, we need to parse it to convert it into a
javascript object

var inputValueParsed = JSON.parse(inputValue);

vals.oTSecurityClearanceLevel = inputValueParsed.oTSecurityClearanceLevel
? inputValueParsed.oTSecurityClearanceLevel : '';
```

```
vals.SupplementalMarking = inputValueParsed.SupplementalMarking ?
inputValueParsed.SupplementalMarking : '';

var holders = [];

var logger = org.slf4j.LoggerFactory.getLogger("debug");

if ((vals.oTSecurityClearanceLevel !== undefined) &&
(vals.oTSecurityClearanceLevel !== ''))

{

holders = JSON.parse(vals.oTSecurityClearanceLevel);

}

if ((vals.SupplementalMarking !== undefined) && (vals.SupplementalMarking
!== ''))

{

//we need to split value, since it is a multivalue attribute

holders = holders.concat(JSON.parse(vals.SupplementalMarking));

//logger.info("holders are: " + JSON.stringify(holders));

//logger.info("**********");

}

outputValue = JSON.stringify(holders);

// enable debug by uncommenting lines below

//logger.info("**********");

//logger.info("inputValue is: " + JSON.stringify(inputValue));

//logger.info("extracted vals are " + JSON.stringify(vals));

//logger.info("**********");

/* NOTE: any code inserted within this block will get overridden if inputs
are regenerated */

/* end of generated script */
```

12.6 OpenText Directory Services fulfillment

To add an application to a user, configure service parameters as required, then in the **Fulfillment Item configuration and mapping** view you must enter the **User Partition ID** within double quotes. This is a mandatory field and must be specified to add the user in OpenText Directory Services. The default script to add an application to a user includes default attributes. If you want to include additional attributes, modify the transformation script for **Content Type** and construct the necessary payload. To create the payload, in the **Content Type** field, click the **{ }**, then transform the script for **http body** and edit the ECMA script as shown in the following examples:

- If the user response value from OpenText Directory Services is in the following format:

  `“description”`: `“Budget Analyst”` or `“urlId”`: `“bolson@efocused.com”`,

  then you must construct the payload as:

  `body.description = “Software Engineer”;//`

  If the value of the description is stored in an attribute from OpenText Identity Governance, then you can add the attribute.

  For example, `body.description = userProfile.displaydescription? userProfile.displaydescription:”;`

- If the user response value from OpenText Directory Services is in the following format:

  ```
  {
  "name" : "sn",
  "values": ["Olson"]
  },
  ```

  Then you must construct the payload as:

  ```
  var values = [];var valuesJson = {};valuesJson.name = 'sn';var snValues
  = [];snValues.push(userProfile.lastName);valuesJson.values =
  snValues;values.push(valuesJson);
  ```

The OpenText Identity Governance OTDS fulfiller supports the following change requests:

- `REMOVE PERMISSION ASSIGNMENT`
- `ADD PERMISSION TO USER`
- `GIVE USER ACCESS TO APPLICATION`
- `REMOVE USER ACCESS TO APPLICATION`
- `REMOVE PERMISSION FROM ACCOUNT`
- `REMOVE ACCOUNT`
- `ASSIGN USER TO ACCOUNT`
- `REMOVE USER FROM ACCOUNT`

NOTE: To fulfill the change request `GIVE USER ACCESS TO APPLICATION` you must configure the partition as an unsynchronized partition.

The OpenText Directory Services fulfiller automatically maps certain mandatory attributes by default. See the following table for the list of mandatory attributes that you must include in the Fulfillment Context attribute.

Fulfillment Context Attributes	Attributes
Recipient	◆ User ID from Source ◆ Last Name ◆ First Name ◆ LDAP Distinguished Name ◆ Provisioning ID
Account	◆ Account ID from Source ◆ Account Disabled ◆ Account Provisioning ID ◆ Account Aliases ◆ Account Name ◆ Account Type ◆ Connected Account Provisioning ID ◆ IDM Account ID
Permission	◆ Permission Type ◆ Permission ID from Source ◆ Permission Name ◆ Provisioning Application Logical ID

13 OpenText REST PAM collectors

This chapter focuses on specific configuration-related information regarding the following collection templates:

- REST PAM Account
- REST PAM Permission

NOTE: The OpenText Privileged Access Manager must have a minimum version of 4.4 and above for you to collect accounts and permissions using the REST PAM collector.

For information about configuring REST PAM templates see the following sections:

- Section 13.1, “Required minimum rights for integration with OpenText Privileged Access Manager,” on page 59
- Section 13.2, “REST PAM account and permission collectors,” on page 59

13.1 Required minimum rights for integration with OpenText Privileged Access Manager

To ensure data is mapped successfully from OpenText Privileged Access Manager to the OpenText Identity Governance catalog, the authorized users must have the required minimum rights in OpenText Privileged Access Manager. A local user or an LDAP user can run the APIs for user roles, resource pools, and assignments. However, the user must be added as a group member or must be mapped to a group and have the `View Access Control Objects` permission in OpenText Privileged Access Manager.

13.2 REST PAM account and permission collectors

OpenText Privileged Access Manager manages and monitors administrative access to servers, networks, and databases to any target application through its access control objects, such as user roles, resources, resource pool, and assignments.

User roles and resource pools are logical groupings, where user roles are allocated permissions to access resources. These resources, in turn, are organized within a resource pool. OpenText Privileged Access Manager utilizes assignments to establish a connection between user roles and the associated resource pool.

The PAM account collector collects unique members and group members from all user roles, and the permission collector collects user roles and members included in the role, resource pool, and the user role-resource pool parent-child relationship. These accounts and permissions are mapped to identities by association or other attributes. Because OpenText Privileged Access Manager uses LDAP as its identity source, the PAM collector maps only LDAP accounts to identities.

When configuring the REST PAM account Collector, configure service parameters as needed, then specify the Account-User Mapping parameter as "`id`" and map it to the identity attribute which holds the `objID`.

Optionally, if you want the PAM accounts to be populated uniquely in the OpenText Identity Governance catalog, then in the **Collect Account** view for Mapped Attributes specify the PAM attributes. For example, ID which is unique to PAM account. Then write an ECMA script for the Collect Account attributes for example:

[`outputValue = "NetiqPAM" + inputValue`]

When configuring the REST PAM Permission Collector, configure service parameters, then depending on the type of permission you want to collect, select the permission type separately for User Role and Resource Pool and specify if you want to collect disabled permissions.

- To map the permissions to an account, specify Permission-Account or User Mapping parameter value as "`ids`" and map it to Account ID.
- To collect the parent-child relationship between User Role and Resource pool, specify the Parent Permission ID value as `parentPermission`.

14 Salesforce collectors and fulfiller

This chapter focuses on specific configuration-related information regarding the following collection and fulfillment templates:

- Salesforce Identity
- Salesforce Account
- Salesforce Permission
- Salesforce Profile Permission
- Salesforce Role Permission
- Salesforce Fulfillment

For additional information about configuring Salesforce templates, see the following sections:

- Section 14.1, "Salesforce collectors," on page 61
- Section 14.2, "Salesforce fulfillment," on page 61

14.1 Salesforce collectors

Using standard OpenText Identity Governance Salesforce collector templates, you can collect data from `User`, `UserRole`, and `Profile` objects. The `User` object is used for Salesforce Identity and Salesforce Account collectors as well as the permission-holder information in the permission collectors.

The generic Salesforce permission collector is configured by default to collect `UserRole` permissions. However, you can configure the collector to collect other permission types such as `UserLicense`, `PackageLicense`, `PermissionSetLicense`, `PermissionSet`, `PermissionSetGroup`, and `Profile`. For your convenience, OpenText Identity Governance also provides Salesforce Role Permission and Salesforce Profile Permission collector templates to collect only UserRole and Profile objects respectively.

14.2 Salesforce fulfillment

The OpenText Identity Governance Salesforce Fulfillment template provides a transformation policy that:

- Executes a query for a single existing user and creates a new Salesforce User if needed
- Assigns or revokes the following permission types: `UserRole`, `Profile`, `PackageLicense`, `PermissionSetLicense`, `PermissionSet`, and `PermissionSetGroup`

To assign some `PermissionSet` or `PermissionSetGroup` permissions, it might be necessary to assign an appropriate license first. We therefore recommend that you assign all licenses before you assign other permission types.

The default transformation policy also includes fulfillment attributes required for fulfillment operations. One required User attribute is **ProfileId**, which must contain the native ID value of a Profile permission. Since all Salesforce Users *must* have a Profile assignment at all times, it is your responsibility to provide a default ID that can be used for new Users or to reset a User whose profile has been removed by OpenText Identity Governance fulfillment actions. This attribute ID should replace the *`ID of default profile`* string in the transformation policy.

Depending on your operations, you might also need to specify additional Fulfillment Context attributes for `userProfile` and `permissionProfile`.

15 SAP collectors

This chapter focuses on specific configuration-related information regarding the following collection and fulfillment templates:

- SAP HR Identity
- SAP User Management Identity
- SAP User Management Account
- SAP User Management Permission

NOTE: OpenText Identity Governance does not currently support SAP collectors in the SaaS environment.

To collect data from an SAP source, OpenText Identity Governance needs the appropriate third-party connector libraries to be installed on the OpenText Identity Governance server.

16 SCIM collectors and fulfiller

This chapter focuses on specific configuration-related information regarding the following collection and fulfillment templates:

- SCIM Identity
- SCIM Account
- SCIM Permission
- SCIM Fulfillment

The System for Cross-domain Identity Management (SCIM) is a protocol for identity exchange, especially across SaaS products. SCIM connectors enable OpenText Identity Governance to integrate with applications seamlessly and support multiple authentication methods.

For additional information about configuring SCIM templates, see the following sections:

- Section 16.1, "SCIM authentication methods and ordinals," on page 65
- Section 16.2, "SCIM collectors," on page 66
- Section 16.3, "SCIM fulfillment," on page 67

16.1 SCIM authentication methods and ordinals

SCIM connectors require a particularly complex configuration template that supports three different authentication types, each of which has different credential parameters that are required to properly configure the collectors and fulfillers. The choice of authentication type and grant type will depend on the use case and what the authentication token endpoint supports.

When using the bearer token authentication method, you can select **Password Flow** (when user involvement is required) or **Client Credential Flow** (for machine-to-machine communication) as the authentication grant type. When using the Client Credential Flow, you will need to specify whether the credentials should be included in the request header or request body .

NOTE: When integrating with other OpenText products using OpenText Advanced Authentication and credential flow authentication method, use `oauth2_aud_base` value as your **Resource** to specify the audience for which the authentication token is being requested. It indicates the API or service that the token is intended to grant access to. The audience typically refers to the unique identifier (such as the URI) of the resource or API you are trying to access.

When using Cloud Bridge, you must also specify a unique ordinal for each authentication method. Use the following table to understand the ordinal number that you need to specify for SCIM authentication methods.

The following table lists the available authentication types and related credentials:

Ordinal (Credential Position)	Authentication Type	Credential Set
3	Basic Auth	◆ User Name ◆ Password
4	Access Token **NOTE:** When the access token expires, replace it with a new access token.	◆ Access Token Header ◆ Access Token
5	Bearer Token	◆ User Name ◆ Password
6	Bearer Token	◆ Client ID ◆ Client Secret

IMPORTANT: For the access token, the user provides the token to connect to the SCIM-compatible application, whereas, for the bearer token, the connector generates the token. When the access token expires, replace it with a new access token.

16.2 SCIM collectors

The SCIM account and permission collectors use unique authentication methods. In addition to specifying the authentication method, you might need to change attribute mapping when configuring the template. SCIM supports singular attributes, complex singular atttibutes, complex multi-valued attributes, and extensions. However, if your application supports any other attributes or extensions different from those mentioned in the SCIM protocol, you can change the attribute mapping in the template by using delimiters. You can use ':' (colon) for attributes, for example, `emails:work:`*`value`*, and '+' (plus) for extensions, for example, `urn:ietf:params:scim:schemas:extension:enterprise:2.0:User+`*`department`*.

To successfully map SCIM accounts and permissions to identities, you must use `email` as the mapping attribute during identity, accounts, and permissions collection. SCIM collects records in batches of up to 999 records, and the default batch collection session timeout value is set to 60 seconds.

By default, the generic SCIM permission collector collects groups as permission for the resource type. However, you can configure the collector to collect other permissions by setting the Resource Type and mapping the attributes of that resource type. For example, if you want to add printers as permission you can give the endpoint of that resource type and map the required attributes to perform the collection.

16.3 SCIM fulfillment

OpenText Identity Governance uses the SCIM fulfillment template for managing identities, and fulfilling change requests for permissions and accounts, especially across SaaS products. Based on the SCIM protocol, the SCIM fulfiller has default attribute mapping that helps you fulfill requests. However, you can change these mappings to match the requirements of your application.

The SCIM fulfiller template allows you to edit the transform script to build the required payload for the change requests for generic fulfillment, user profiles, permissions, and accounts. The ECMA script includes comments that guide you through the payload generation process. After you generate the payload, OpenText Identity Governance sends the payload for fulfillment. The SCIM fulfiller generates the payload for the following change requests:

- `ADD_APPLICATION_TO_USER`
- `ADD_PERMISSION_TO_USER`
- `REMOVE_ACCOUNT_PERMISSION`
- `REMOVE_PERMISSION_ASSIGNMENT`
- `REMOVE_ACCOUNT`

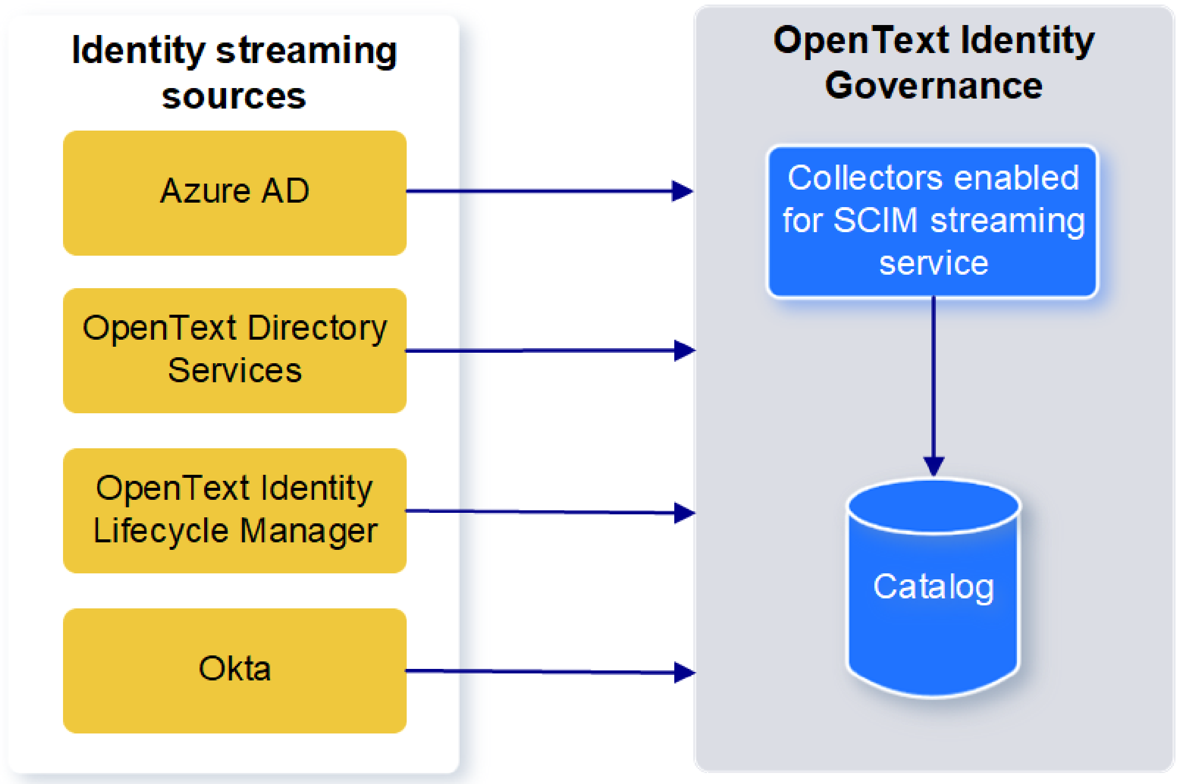
NOTE: Each application you integrate with might not support all the above change requests.

17 SCIM streaming collector

This chapter focuses on specific configuration-related information regarding the SCIM streaming collector.

Multiple identity providers, such as Azure and OKTA, recommend or require integrators to support SCIM push capabilities to get identity information from them. Other identity providers, such as OpenText Directory Services, have the ability to push SCIM data to a consumer.

Figure 17-1 *SCIM push overview*


Identity streaming sources
Azure AD
OpenText Directory Services
OpenText Identity Lifecycle Manager
Okta
OpenText Identity Governance
Collectors enabled for SCIM streaming service
Catalog

OpenText Identity Governance provides a way for an administrator to create, modify, and delete streaming identity sources. Streaming identity sources are non-mergeable identity sources that receive identities and groups through SCIM push requests. Each streaming identity source needs a unique ID and is associated with a service account. The unique ID allows the application to route incoming SCIM requests to the appropriate data source. When enabled, the Advanced Entity Viewer displays the synchronized users and groups on the data source page. Authorized administrators can also view users and groups from the streaming source in the identity catalog. Administrators can configure the Catalog settings to display the Streaming Identity Source column.

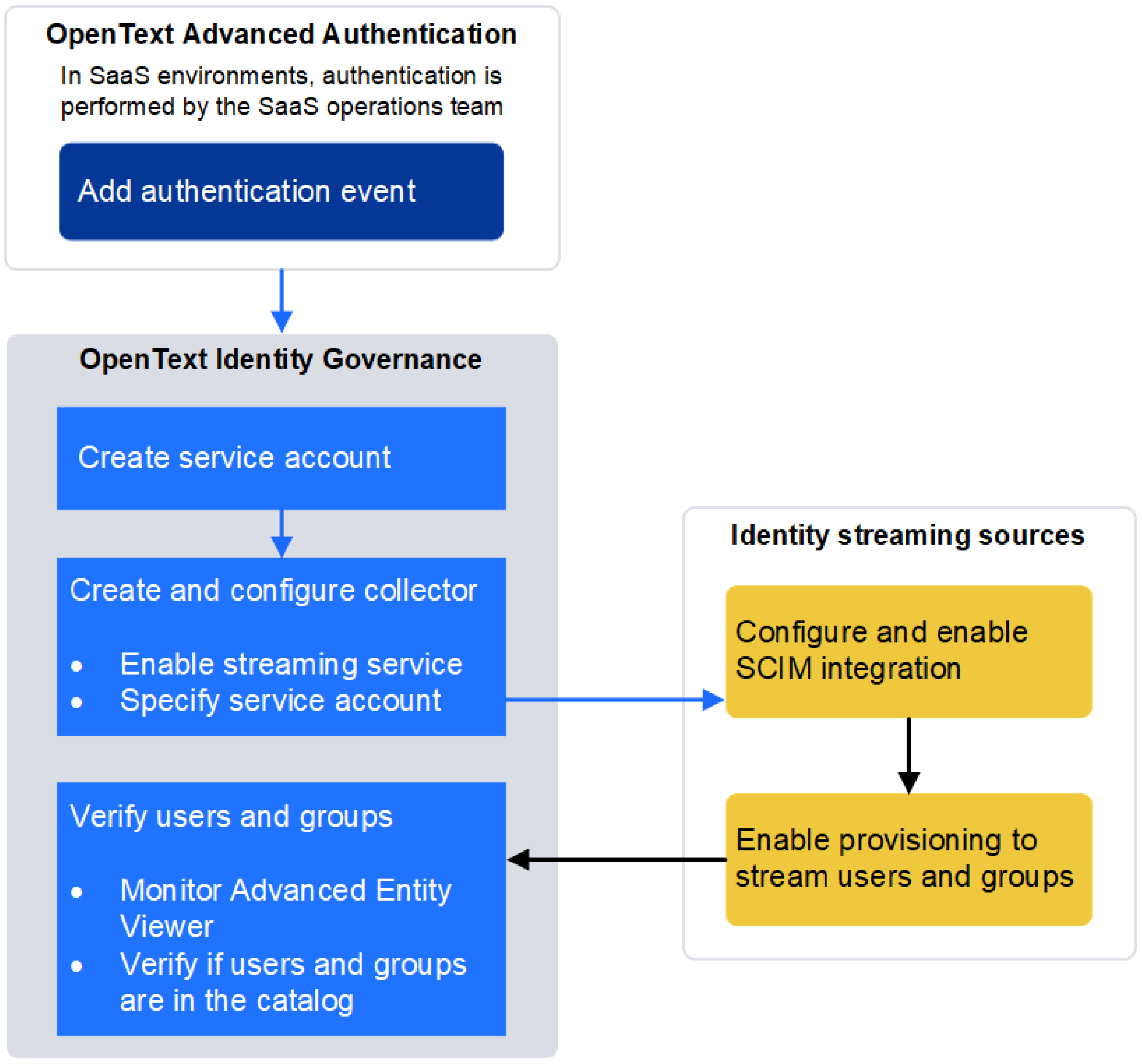
OpenText Identity Governance provides the ability to receive identity and group information from identity providers through the SCIM protocol specified in RFC 7643 and RFC 7644. Not all of the features of the RFCs are supported - only those listed below that are necessary to interact with the primary SCIM providers, such as OKTA, Azure, Microsoft Entra ID, OpenText Directory Services, and OpenText Core Identity Lifecycle Manager that OpenText Identity Governance supports:

- Defining streaming identity sources to handle incoming SCIM requests using the SCIM Streaming collector template
- Processing of incoming SCIM requests including:
  - Creation of Users and Groups using POST (RFC 7644, section 3.3)
  - Modification of Users and Groups using PUT (RFC 7644, section 3.5.1) and PATCH (RFC 7644, section 3.5.2).
  - Deletion of Users and Groups using DELETE (RFC 7644, section 3.6)
  - Retrieval of individual Users and Groups using GET (RFC 7644, section 3.4.1).
  - Query for Users and Groups using GET (RFC 7644, section 3.4.2) or POST (RFC 7644, section 3.4.3)
  - Bulk update requests using POST (RFC 7644, section 3.7)
  - Service provider requests using GET (RFC 7644, section 4)
- Mapping of attributes from the following three primary SCIM schemas:
  - `urn:ietf:params:scim:schemas:core:2.0:User.` This is the core SCIM User schema (RFC 7643, section 4.1)
  - `urn:ietf:params:scim:schemas:core:2.0:Group`. This is the core SCIM Group schema (RFC 7643, section 4.2).
  - `urn:ietf:params:scim:schemas:extension:enterprise:2.0:User`. This is the enterprise User schema extension (RFC 7643, section 4.3).

  Not all attributes from these schemas are mapped by default. If you want to map additional attributes, it may be necessary to create extended OpenText Identity Governance attributes and map the additional SCIM attributes to those extended attributes. Mapping of attributes from custom SCIM schemas would likely require the same process.
- Automatic detection of SCIM events

Find next a high-level overview of the SCIM streaming service integration and configuration process:

Figure 17-2 *SCIM streaming service and OpenText Identity Governance integration and configuration process*


OpenText Advanced Authentication
In SaaS environments, authentication is performed by the SaaS operations team
Add authentication event
OpenText Identity Governance
Create service account
Create and configure collector
• Enable streaming service
• Specify service account
Verify users and groups
• Monitor Advanced Entity Viewer
• Verify if users and groups are in the catalog
Identity streaming sources
Configure and enable SCIM integration
Enable provisioning to stream users and groups

For more information about enabling streaming identity sources, see the following topics:

- Section 17.1, "SCIM streaming authentication," on page 72
- Section 17.2, "Configuring Azure AD for the SCIM streaming service," on page 72
- Section 17.3, "Configuring Okta for the SCIM streaming service," on page 73
- Section 17.4, "Configuring OpenText Directory Services for the SCIM streaming service," on page 74
- Section 17.5, "Configuring OpenText Core Identity Lifecycle Manager for the SCIM streaming service," on page 75
- Section 17.6, "Configuring the SCIM streaming collector," on page 77

17.1 SCIM streaming authentication

SCIM streaming authentication process begins with authorized administrators creating an event in OpenText™ Advanced Authentication using a unique client ID and client secret. The same client ID is then used to create a service account in OpenText Identity Governance. This service account allows the external identity source to interact securely with OpenText Identity Governance. To enable secure communication with identity streaming sources, you must specify the following credentials:

Identity streaming source	Credential set
Azure AD and Okta	Bearer token generated with: ◆ Client ID ◆ Client secret ◆ Access Token URL **NOTE:** The validity of the bearer token can be configured for a maximum of 10 minutes.
ILM and OTDS	◆ Client ID ◆ Client secret ◆ Access Token URL

17.2 Configuring Azure AD for the SCIM streaming service

Before you start your configurations, ensure that you have an Azure AD premium account with administrator permissions:

To configure Azure AD:

1 Create a new application.

- **1a** Log in to Azure Portal with your Azure AD admin credentials.
- **1b** Select **Microsoft Entra ID**.
- **1c** From the left panel, select **Manage** and expand it.
- **1d** Select **Create your own application**.
- **1e** Specify a name for the application and choose to integrate any other application you do not find in the gallery.
- **1f** Select **Create**.

2 Enable SCIM provisioning.

- **2a** On the All Applications page, select the application name.
- **2b** On the Properties page, from the left panel, select **Manage** and expand it.
- **2c** Select **Provisioning**.
- **2d** On the Overview page, from the left panel, select **Manage** > **Provisioning**.
- **2e** In the **Admin Credentials** section, type the SCIM Endpoint URL of the OpenText Identity Governance server and the unique identifier. Use the following format:

```
https://igurl/api/scim/UNIQUE_IDENTITY_SOURCE_ID
```

2f Generate the bearer token using a script or a tool such as Insomnia, then enter the token in the **Secret Token** field.

2g Test the connection to ensure that it is successful.

3 Configure attribute mapping to map user attributes between Azure AD and OpenText Identity Governance.

3a (Optional) In the **Mappings** section, click the provision link for users and groups to edit the default mappings.

Azure AD provides predefined mappings for common attributes such as `userPrincipalName`, `email`, `firstName`, or `lastName`.

3b (Optional) Click **Add New Attribute**.

4 Provision users on demand.

4a Select **Provision on demand** for manual provisioning.

4b Search for an existing user or group and select the user.

4c Click **Provision**.

To validate whether users and groups are provisioned, log in to OpenText Identity Governance and navigate to **Catalog** > **Identities** or **Groups**.

NOTE: When provisioning users or groups, the process can take up to 40 minutes, while the bearer token is valid for only 10 minutes. Hence, we do not support automatic mode of provisioning.

17.3 Configuring Okta for the SCIM streaming service

Before you start your configurations, ensure that you have full administration access in Okta. For more information, see Okta developer website.

To configure Okta for the SCIM streaming service:

1 Create a SCIM integration in Okta.

1a Log in to the Okta administration console with your admin credentials.

1b Create a new SCIM app in Okta.

1b1 On the Okta dashboard, from the left panel, select **Applications** > **Applications**.

1b2 Select **Browse App Catalog**.

1b3 Search for `SCIM 2.0`, then select SCIM 2.0 (OAuth Bearer Token) from the list of available templates.

1b4 Select **Add Integration** and ensure that the general settings have the required default selections.

1b5 Click **Next**, then **Done**.

2 Configure the SCIM API integration.

2a On the new App page, select the **Provisioning** tab.

2b Select **Configure API Integration** > **Enable API Integration**.

2c Type the SCIM Endpoint URL of the OpenText Identity Governance server and the unique identifier that you specified in Step 6. Use the following format:

`https://`*`igurl`*`/api/scim/`*`UNIQUE_IDENTITY_SOURCE_ID`*.

2d Generate the bearer token using a script or a tool such as Insomnia, then enter the token in the **OAuth Bearer Token** field.

3 Test the SCIM connection.

3a Click **Test API Credentials** to verify the connection between Okta and your SCIM server.

3b Save your provisioning settings.

4 Enable SCIM provisioning.

4a On the App page, from the **Settings** panel, select **To App** and then **Edit**.

4b Enable the necessary provisioning actions from the available common operations to:

- Create users
- Update user attributes
- Deactivate users

NOTE: The current Okta integration does not support group provisioning.

4c Save your settings.

5 Provision users.

5a From the left panel, select **Directory** > **People**.

5b Add a new user by specifying all the necessary fields.

5c Click **Save**.

5d Select the new user that you added.

5e Click **Assign Applications**.

5f Assign the application that you created in Step 1.

5g Click **Save**.

5h Click **Done**.

To validate whether users are provisioned, log in to OpenText Identity Governance and navigate to **Catalog** > **Identities**.

17.4 Configuring OpenText Directory Services for the SCIM streaming service

SaaS Operations members and authorized administrators must work together to configure OpenText Directory Services and create service accounts.

For detailed procedures for installing and configuring OpenText Directory Services, refer to the *OpenText Directory Services Installation and Administration Guide* on the OpenText Directory Services documentation website (https://developer.opentext.com/ce/products/opentext-directory-services/documentation/directory-services-otds). You will need a My Support account to access the documentation.

The following list provides a high-level view of the actions authorized administrators must perform to push identities to OpenText Identity Governance:

- Create partitions.
- Create a resource and specify SCIM 2.0 connection parameters as follows:
  - For the base URL, append the unique ID that was specified in the SCIM streaming collector.
  - Specify the OAuth Token URL, Client ID, and Client Secret. This will be used as the service account Client ID and Client Secret.
  - Save the resource settings to trigger the automatic creation of an access role.
- Include groups on the Access Role tab, and on the Access Role Details page add the partitions that were previously created.
- Add new users and groups on the Partitions tab.

After the initial configuration, all users and groups will be automatically pushed to OpenText Identity Governance. Authorized OpenText Identity Governance administrators can then view the new users and groups on the SCIM Streaming identity data source page (using Advanced Entity Viewer) or on the Catalog page.

17.5 Configuring OpenText Core Identity Lifecycle Manager for the SCIM streaming service

Before you start your configurations, ensure that your SCIM server supports:

- SCIM 2.0 and handles user and group provisioning operations, including create, update, and delete actions
- Service account-based authentication for Client ID and Client Secret

To install and configure the SCIM driver:

1. Log in to the OpenText Core Identity Lifecycle Manager administration console with your admin credentials.
2. Create a new SCIM driver.
   - **2a** Navigate to the Driver Management section.
   - **2b** Select **Driver Configuration**.
   - **2c** Click the + sign and search for SCIM.
   - **2d** Expand **Tools** and click the **+** sign next to SCIM-Driver for SCIM.
   - **2e** Select the following:
     - In Select Driver Base Configuration, select **SCIM Base ILM** and download the missing packages, then click **Next**.
     - In Select Mandatory Features, select **SCIM Default Package** and download missing packages, then click **Next**.
     - In Select Optional Features, select only the package names that are required for OpenText Core Identity Lifecycle Manager and OpenText Identity Governance Integration, such as **SCIM JSON Package**, **SCIM ILM IG Integration**, and **SCIM Default Filter Mapping ILM**. Download the missing packages, then click **Next**.

2f In Required Package Dependencies, click **Next**.

2g In Driver Information, specify the driver name, then click **Next**.

2h In Driver Configuration, select **Connection** > **Remote Loader**, then click **Next**.

2i Set Publisher Options, set **Enable Publisher** channel to **No**, then click **Next**.

3 Configure connection parameters.

3a Select **Configuration** from the left panel.

3b Set Authentication Method to **OAuth2.0**.

3c Select **Bearer** for OAuth2.0 Token Management.

3d In the Access Token URL field, specify the access token URL from OpenText Identity Governance.

3e In the Query Options area:

3e1 Type the name as `grant_type` and value as `client_credentials`.

3e2 Type the name as `client_ID` and value as the service account client ID.

3e3 Delete the issuer.

3f In the Secret Query Options area:

3f1 Type the name as `client_secret` and the value as the service account client secret.

3f2 Delete the refresh token.

3g In the Header Fields area type:

3g1 Name as `Accept` and value as `application/JSON`.

3g2 Name as `Content-Type` and value as `application/x-www-form-urlencoded.`

3h In the Application Truststore File field:

3h1 (Conditional) If OpenText Identity Governance uses `https`, specify the location of the OpenText Core Identity Lifecycle Manager server where the OpenText Identity Governance CA certificates are mounted.

3h2 (Conditional) If OpenText Identity Governance uses `http`, keep the field blank.

3i In Schema Settings, select 2.0 as the Schema Options. Type the SCIM Endpoint URL of the OpenText Identity Governance server and the unique identifier. Use the following format:

`https://`*`igurl`*`/api/scim/`*`UNIQUE_IDENTITY_SOURCE_ID`*

4 Start the driver.

4a Wait for the SCIM driver to be installed. Then start the driver and wait for the remote loader to start.

5 Provision users and groups.

5a Log in to OpenText Core Identity Lifecycle Manager, then navigate to **Administration** > **Users**.

5b Add a new user.

5c To add groups, navigate to **Administration** > **Groups**.

5d Add a new group.

To validate whether users and groups are provisioned, log in to OpenText Identity Governance and navigate to **Catalog** > **Identities** or **Groups**.

17.6 Configuring the SCIM streaming collector

Before configuring the template, ensure that you have service account credentials and have configured the streaming service provider such as Azure AD, Okta, OTDS, and OpenText Core Identity Lifecycle Manager as outlined in the following sections:

- Section 17.2, "Configuring Azure AD for the SCIM streaming service," on page 72
- Section 17.3, "Configuring Okta for the SCIM streaming service," on page 73
- Section 17.4, "Configuring OpenText Directory Services for the SCIM streaming service," on page 74
- Section 17.5, "Configuring OpenText Core Identity Lifecycle Manager for the SCIM streaming service," on page 75

To configure one or more streaming identity sources to receive SCIM data:

1. In OpenText Identity Governance, select **Data Sources** > **Identities**.
2. Add an identity source.
3. Type a name and description.
4. Select **Publish without merging** as the publish behavior.
5. Enable the identity source as **Streaming identity source**.
6. Type a unique identity source identifier.

   NOTE: This unique ID will be part of the base path in the URL that is used to send SCIM data to OpenText Identity Governance. This unique ID is extracted from the path to determine the identity source for which the SCIM data is intended.

7. Type the streaming collector name.
8. Specify the service account.
9. (Optional) Deactivate the capability to delete users received from streaming identity sources.
10. Define mappings from SCIM content to OpenText Identity Governance User and Group attributes.
11. Click {} and define transformation scripts.
12. Save your configurations.

18 ServiceNow collectors and fulfillers

This chapter focuses on specific configuration-related information regarding the following collection and fulfillment templates:

- ServiceNow Identity
- ServiceNow Account
- ServiceNow Permission
- ServiceNow Generic Fulfillment
- ServiceNow Incident Fulfillment
- ServiceNow Request Fulfillment
- ServiceNow Task Fulfillment

For information about configuring ServiceNow templates, see the following sections:

- Section 18.1, "ServiceNow collectors," on page 79
- Section 18.2, "ServiceNow fulfillment," on page 79

18.1 ServiceNow collectors

Collection from the ServiceNow portal is similar to other identity and application data sources. When you are configuring the templates, we recommend that you map the User ID from Source, Account ID from Source, and Permission ID from Source attributes to a unique identifier, for example, `sys_ID` as the mapping attribute. After collecting data, OpenText Identity Governance maps accounts and permissions to identities by association or other attributes.

18.2 ServiceNow fulfillment

OpenText Identity Governance provides four distinct ServiceNow fulfillment target types: Generic, Incident, Request, and Task. Whereas the OpenText Identity Governance fulfillment targets are meant to provision or deprovision an application or permission to a user, the ServiceNow fulfillment types have a different purpose. Depending on your organization's requirements, select the template from the list that best suits your needs:

- Select the *generic* fulfillment type if you need customization that the other fulfillment types allow. In the out-of-the-box scenario, the generic fulfillment type generates an incident. It takes information from the OpenText Identity Governance change request item and then uses an ECMA script to create a SOAP XML payload which is then sent to ServiceNow.
- Select the *incident* fulfillment type to create a ServiceNow incident to process each change request item. Fulfillment verification happens at the incident level.

- Select the *request* fulfillment type to create a ServiceNow request to process each change request item. Fulfillment verification happens at the request level.
- Select the *task* fulfillment type to group fulfillment items together based on the user. It creates a hierarchy of ServiceNow request, request items, and tasks. Here one request contains one to n request items grouped by user. Inside each request item, there are tasks for the actual change request item. Fulfillment verification happens at the task level.

The fulfillment status from these fulfillment types remain in the pending verification state. After the request is approved, the status in OpenText Identity Governance changes to verified.

19 SharePoint collectors

This chapter focuses on specific configuration-related information regarding the following collection and fulfillment templates:

- SharePoint Identity
- SharePoint Account

Microsoft SharePoint is a browser-based collaboration and document management tool that allows administrators to grant specified access rights to individual users and groups.

To gather information from SharePoint, the Service Account you use to configure the SharePoint collection must be a member of the WSS_ADMIN_WPG local group on the SharePoint server.

NOTE: OpenText Identity Governance does not support using the SharePoint collector for SharePoint Online.

20 Workday collectors and fulfiller

This chapter focuses on specific configuration-related information regarding the following collection templates:

- Workday Identity
- Workday Account
- Workday Permission
- Workday Fulfillment

Before configuring these templates, create an integration account and ensure that the minimum rights required to integrate with Workday systems are assigned to the integration groups and users in the Workday application.

For additional information about configuring Workday templates, see the following sections:

- Section 20.1, “Required minimum rights for integration with Workday,” on page 83
- Section 20.2, “Workday collectors,” on page 84

20.1 Required minimum rights for integration with Workday

The three minimum security domain rights that must be assigned to the integration group and users to get the data necessary for the default mappings in the Workday Identity Collector are:

- Person Data: ID Information
- Worker Data: Public Worker Reports
- Workday Accounts

The following rights are required to collect the necessary data for the default mappings in the Workday Application Collector:

- Account collector
    - Workday Accounts
    - Worker Data: Public Worker Reports
- Permission collector
    - Manage: Organization Roles
    - Org Designs: Assign Roles
    - User-Based Security Group Administration
    - Manager: Organization Integration

20.2 Workday collectors

Security groups control access to data in Workday. Security groups are a collection of users or of objects that are related to users. OpenText Identity Governance provides default templates for the Workday account and permission collections. Workday permission collectors support two types of permission collections: User Based Security Group and Role Based Permissions. Role-based permissions are always associated with a specific organization. When using role-based permission collectors, you can also collect permission hierarchy. Collected role-based permission in the catalog includes role name, permission, and organization as the name of the permission, and displays permission relationships.

When configuring the Workday Account Collector, configure service parameters as needed, then specify the Account-User Mapping parameter as `WorkdayUserName` and map it to `Object GUID` to join accounts to identities.

When configuring the Workday Permission Collector, configure service parameters, then select the permission type.

- To collect user-based security group permissions, specify the Permission-Account or User Mapping parameter value as `WorkdayUserName` and map it to `Account Name` to join permissions to the account.
- To collect role-based permissions, specify the Permission-Account or User Mapping value as `WorkforceID` and map it to `Workforce ID` to map permissions to identities. Additionally, leave the organization type blank to collect all role-based permissions or specify an organization type to collect permissions associated with an organization.

  When specifying a specific organization, to collect the hierarchy of role-based permissions using the organization hierarchy, map the Parent Permission ID to `wd-superior_organization`. Mapping this will collect and establish the child/parent permission relationship for role-based permissions.

21 REST generic fullfiller

This chapter focuses on specific configuration-related information regarding the REST Generic fufiller.

The REST Generic fulfiller supports Oauth 2.0 and has three different authentication types: basic, generate bearer token, and enter access token. When using the bearer token authentication method, you must specify the username and password, then the OAuth2 client ID and secret for API access to the target application. The process for configuring the applications and generating the client ID and secret will vary depending on your target application. For additional information about getting the client ID and secret, contact the application owner.

Log in to the Cloud Bridge URL, then specify the ordinal when adding credentials.

Ordinal (Credential Position)	Authentication Type	Credential Set
3	Basic Auth	◆ User Name ◆ Password
4	Access Token	◆ Access Token Header ◆ Access Token
5	Bearer Token	◆ User Name ◆ Password
6	Bearer Token	◆ Client ID ◆ Client Secret

OpenText Identity Governance uses the REST Generic fulfiller for fulfilling requests for any REST-based application using REST endpoints. This fulfiller also supports OAuth 2.0. The REST Generic fulfillment template allows you to customize the template. While configuring the **Fulfillment Item configuration and mapping,** click **{..}** for **Content**, then specify the `service_method` and the `http_body`.

22 Workflow Service fulfiller

This chapter focuses on specific configuration-related information regarding the Worlflow Service fulfiller.

OpenText Identity Governance uses the Workflow Service fulfillment target to get a workflow from the Workflow Service and run the workflow to fulfill changesets. You can either use an existing workflow or create a workflow in OpenText Identity Governance. OpenText Identity Governance then sends the `changeitemid` to the Workflow Service to process the fulfillment.

NOTE: To edit the workflow, click the **Edit** link next to the **Workflow** field to launch the Workflow Builder in the Workflow Administration Console. In the Workflow Builder, ensure that the default IGA fulfillment request form is selected for the fulfillment request to complete. Using any other form for your fulfillment request might result in unpredictable behavior.

The Workflow Service identifies the entity, parses the information, and completes the task. The Workflow Service, however, does not inform OpenText Identity Governance when the task finishes. To check the fulfillment steps or fulfillment status, select **Fulfillment** > **Status** or **Requests** > **Requests**.

A Supported OpenText Identity Manager drivers and packages

OpenText Identity Governance provides IDM entitlement application definition and application templates to collect account and permission entitlements from an on-premises OpenText Identity Manager environment. To successfully collect all accounts and permissions, the supported drivers must be running. Find below a list of the supported drivers.

- Drivers in OpenText Identity Manager 4.9 (https://www.netiq.com/documentation/identity-manager-49-drivers/) and later patched versions
- OpenText Identity Governance Assignment collection: MFIGASGMTCOL_1.0.0.20220110104142

Driver	Minimum Driver Version	Minimum Package Version
Active Directory	4.1.3.0500	• NOVLADENTEX_2.5.8.20230710122417
Azure AD	5.2.0.0000	• MFAZUREENTL_1.0.4.20220913175019 • MFAZUREXROLE_1.0.2.20211125114229
Bidirectional	4.1.0.0000	• NOVLEDIR2ENT_2.2.7.20211118165416
Groupwise REST	4.1.0.0000	• NOVLGRPWRAEN_3.1.1.20211209173838
JDBC	4.2.2.0000	• NOVLJDBCBISN_2.0.0.20211208134901 • NOVLJDBCENTI_2.4.4.20211208135336 • NOVLORAINSYN_2.1.1.20211222134359 • NOVLSQSIDSYN_2.1.2.20221213111655 • NOVLPGSINSYN_2.1.1.20211220124959
Lotus Notes	4.1.2.0.0000	• NOVLNOTEENT_2.4.1.20211118113748

Driver	Minimum Driver Version	Minimum Package Version
SAP User Management	4.1.0.0000	• NOVLSAPUFENT_2.3.6.20240307114543 • NOVLSAPUMIG_1.0.0.20211217153953
SCIM	1.1.0.0000	• NETQSCIMENT_1.0.1.20211223151040 • NETQSCIMBASE_1.0.2.20240226003348
Workday	1.4.0.0000	• NETIQWDENT_1.0.0.20210505165701