CyberArk SCIM Server Implementation Guide

Version 1.3.0 and above

Important Notice

Table of Contents

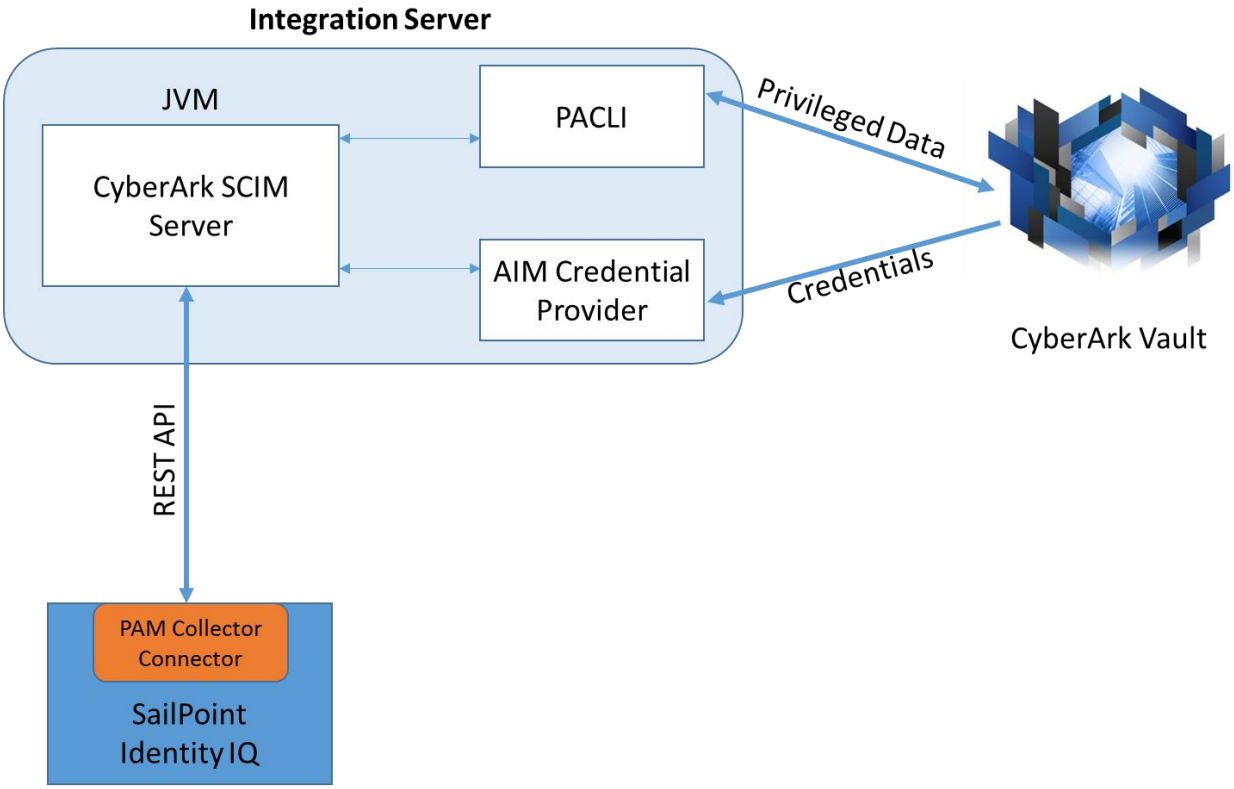
Introduction

The CyberArk SCIM server is a Java application conforming to the System for Cross-domain Identity Management (SCIM) RFC7644 Standard when querying and modifying Users and Groups. When querying and modifying Accounts, Safes and Permissions the SCIM server adheres to the guidelines of the SCIM Extension for Privileged Access Internet-Draft (work in progress).

https://tools.ietf.org/html/rfc7644

https://tools.ietf.org/html/draft-grizzle-scim-pam-ext-01

This capability allows an Identity Provider like SailPoint to query and modify Privileged Data such as Users, Groups, Accounts, Safes, and Permissions through a Web Services interface (REST API). The SCIM framework relies on CyberArk PrivateArk Command Line Interface (PACLI) to query and update privileged data from the CyberArk Vault. The AIM Credential Provider is used to retrieve account and login information:



CyberArk SCIM Requirements

System Requirements

Suggestion is **at least**:

- 8 Core Processors
- 32GB of RAM
- Microsoft Windows Server 2012 R2
  - Server 2016 and 2019 are supported
- Java Virtual Machine version 1.8
  - OpenJDK is supported

HA Suggestions

- Active/Passive
- Active/Active is possible but could produce more problems than it solves. The SCIM server works mostly off of cache to cut down on the Vault traffic and the NLB could cause the cache to become invalid. Some type of Session Persistence/Affinity is required.
- If you have questions, please reach out to CyberArk Support for guidance.

CyberArk Requirements

- CyberArk Vault Server v9.x or higher
- AIM Credential Provider v9.x or higher

SailPoint Requirements

- Identity IQ v7.1 patch 2
- PAM Module

Pre-Installation

Stand-Alone SCIM

This document will outline two separate installation types, the first will guide you through the setup of the **CyberArk Integration Server**. Once this portion of the installation is complete this will provide a working SCIM server ready to integrate with any Identity Provider solution.

- AIM Credential Provider (CP) for Windows
- PACLI
- Java version 1.8 (64bit highly suggested)
- Define the CyberArk-SCIM Application ID
- Modify configuration files to achieve desired workflow

SCIM Cache

The cache is the main feature and most important function of the CyberArk SCIM. It is a permissions mapping of the entire Vault translated into a readable, standard format to be queried by an Identity Governance platform. Every User, Group, Safe, and Object with a mapping of Users **to** Groups and the permissions of every User and Group **to** all Safes and subsequent "PrivilegedData" in each Safe. SCIM **does not support OLAC** at this time, and it is not on the roadmap.

The cache is built by querying the Vault in groupings of the information collected and once a grouping has been considered "invalidated" it is removed and rebuilt. The cache becomes invalidated by either the cacheRefresh value expiring or the removeAllCacheBeforeRefresh parameter set to true and something triggers a refresh. Below are the groupings that make the SCIM cache:

- **User**
- **Group**
- **Containers** (Safes)
- **ContainerPermissions** (Safe Ownership)
- **PrivilegedData** (Account Objects)

SailPoint Identity IQ

The second portion will continue with the implementation of the PAM Module into **SailPoint Identity IQ** (IIQ).

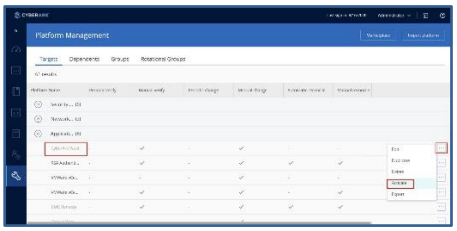
- PAM Module
- SailPoint Application ID
- Confirm necessary XML files

Pre-Installation Tasks

Enable the CyberArk Vault platform:

1. Login to the PVWA as the Administrator user

2. Click the **Administration** Tab
3. Click **Platform Management**
4. Select the **CyberArk Vault** platform
5. Under **Status**; Select **Active**
6. **Save**; by clicking the disk icon



Download and Install Java

JRE is the bare minimum needed as SCIM needs JVM to function. We highly suggest using the 64-bit version to allow for more granular control of how the JVM will perform and consume system resources.

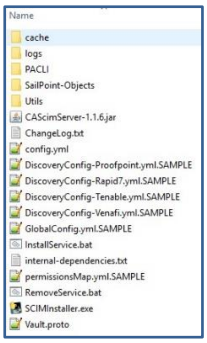
1. Download Java
2. Run Java install
3. It is recommended to set a **%JAVA_HOME%** path in the server's **Environment Variables** if the Java install did not do that as part of the installation

Inventory of the SCIM installation package

1. Download the SCIM distribution zip file from the **CyberArk MarketPlace**.
2. Extract the contents from the zip file
3. Copy the **CyberArk-SCIM** folder to the root C: drive

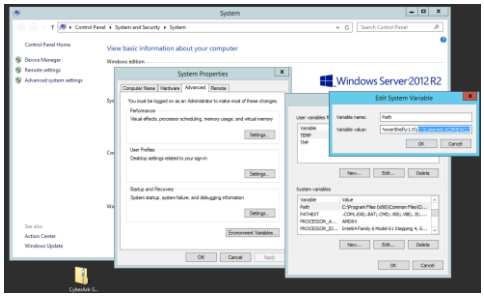
In the CyberArk SCIM folder, there should be the following files and folders:

Files		**Folders**
CAScimServer-<version>.jar	ChangeLog	cache
Config.yml	GlobalConfig.yml.SAMPLE	logs
InstallService.bat	internal-dependencies.txt	PACLI
permissionsMap.yml.SAMPLE	RemoveService.bat	SailPoint-Objects
SCIMInstaller.exe	Vault.proto	Utils
DiscoveryConfig<vendor>.yml.SAMPLE		**Additional Configurations**



Add Environment Variables

1. Open the Start Menu; Right-Click This PC and Select **Properties**
2. In the System window; Select **Advanced system settings**
3. Select the **Advanced** Tab, and Click **Environment Variables**
4. In System Variables Select Path; Click **Edit**
5. In the Edit System Variable window append the path of PACLI to the Variable value:
   - **C:\CyberArk-SCIM\PACLI**
6. If the Java install did not add a **%JAVA_HOME%** entry, now is a good time to do it:
   - **C:\Program Files\Java\bin**



Pre-installation checklist

Confirm you have the following items prior to running the installation script:

- A Vault Administrator user with the following User Rights. Default is set to **Administrator**:

  Add Safes
  Audit Users
  Add/Update Users

- Your Vault server's **address** and **port** number.
- The name of your Vault CPM user. Default is set to **PasswordManager**.
- The password for your client or IAM governance user (i.e. **client-user** or **SailPoint-User**)

  See the Verify the Installation section for a description
- A comma separated **(with no spaces in between)** list of groups to give your **SCIM-user** privileges to provision users/groups/accounts to a safe. If you don't have this list, you can hit return to leave it blank then add the **SCIM-user** to these groups post installation. We recommend you at least provide **Auditors**, then add extra groups if necessary. The **SCIM-user** is the account that OOB will conduct all SCIM functions and if you are going to be provisioning from an IAM Governance platform we suggest providing **Vault Admins** as well.
- The set of SCIM Server permissions you want to assign to the IAM Governance platform user.

  See SCIM Server permissions for details

Additional Configurations

The Additional Configurations folder houses some configurations enables SCIM to either work around issues that can be presented or add additional features that were not provided in versions past. Here we will go into detail on what has been provided

CAScimServer-11.x-1.3.0.jar: This is the SCIM jar file that is to be used if you have not upgraded to 12.1. This supports all versions of Credential Provider (CP) prior to 12.1, but still have the logic modifications for this release

DiscoveryConfig-<partner>.yml.SAMPLE: These are the same Discovery Configuration files present in previous releases, but as we continue to add Partners and their capabilities for Discovery, we placed them here for the sake of organization.

InstallService_char.bat: We have seen environments with characters that conflict with the character set that comes OOB and a way to force SCIM to conform to a character set is by exclusively setting it in the creation of the service. This bat file has parameters to exclusively set the character set.

InstallService_nssm.bat: This is a bat file that will install the SCIM service with the functionality of printing to 2 additional log files **service.log** and **error.log.**

- **Service.log:** Provides additional information on the HTTP status code on incoming requests
- **Error.log:** Provides output on errors that nssm is experiencing when running as a Windows service.

syslog_enabled_config.yml.SAMPLE: This is a config.yml that contains a framework in place that will send CAScimServer logs to a syslog receiver. This can be modified to send any of the logs to a receiver of your choice.

- **host:** The hostname or IP of the receiving syslog solution
- **port:** Port that the syslog receiver will be listening on
- **facility:** This allows for you to assign the messages sent to the facility of your choice. CyberArk CorePAS Vault syslog leverages **local0**
- **threshold:** The lowest level of events to write to the file.

- **stackTracePrefix:** The prefix to use when writing stack trace lines (these are sent to the syslog server separately from the main message)
- **logFormat:** The Logback pattern with which events will be formatted. See the Logback documentation for details.

For more information on configuring Syslog in the logging portion of the Config.yml please access the following link:

https://www.dropwizard.io/en/latest/manual/configuration.html#syslog

Below is an excerpt from the **syslog_enabled_config.yml.SAMPLE** file:

```
    com.cyberark:
      level: DEBUG
      additive: false
      appenders:
        - type: file
          currentLogFilename: .\logs\CAScimServer.log
          archivedLogFilenamePattern: .\logs\CAScimServer-%d.log.gz
          archivedFileCount: 5
          logFormat: "%-5level|%d{YY/MM/dd
HH:mm:ss.SSS}|%class{0}|%method|%line|%thread|%replace(%msg){'[\r\n]+
','  '}%n"
          includeCallerData: true
        - type: syslog
          host: 1.1.1.1
          port: 514
          facility: local0
          threshold: ALL
          stackTracePrefix: \t
          logFormat: "%-5p [%d{ISO8601,UTC}] %c: %m%n%rEx"
```

Installation

- Installing the Credential Provider
  - Running the Installer
  - Adding the Provider User as an Owner to the SCIM Config Safe (**Pov_<hostname>**)
  - Adding the **CyberArk-SCIM** applicationID to the SCIM Config safe
- Installing SCIM
  - Running the SCIM Installer
  - Verify the Installation

IMPORTANT NOTES!

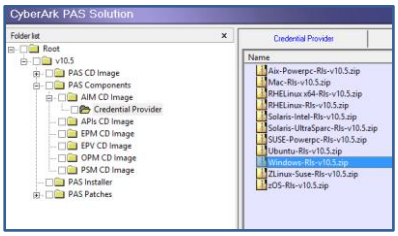
Ensure that the installation of the Credential Provider (CP) runs successfully, if issues installing the CP occur, open a case with CyberArk Support and get this resolved prior to continuing with the installation.

The SCIM installer can only be run **once**, if you aborted in the middle of the install or you have already run it once, you will need to follow the uninstall instructions in order to install again. The uninstall instructions can be found at the end of this document in the Uninstall Procedure section.

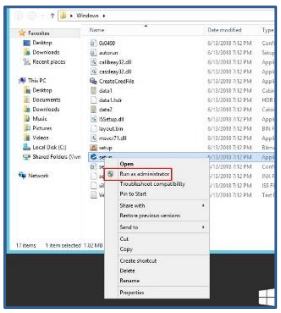
Installing the Credential Provider

Running the Installer

Download the Windows install zip file for the Credential Provider from the **Support Vault** to the SCIM Server



1. Unzip the install package
2. In the installation folder Right-Click the **setup** file and **Run as Administrator**



Installing SCIM

Running the SCIM Installer

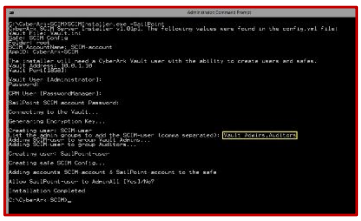
1. Open an administrative command window and navigate to **C:\Cyberark-SCIM**
2. Run the following command:

   SCIMInstaller.exe -SailPoint

   NOTE: If you're installing the SCIM server for SailPoint Identity IQ, add the **"–SailPoint"** flag. The "-SailPoint" syntax is case sensitive
3. Alternatively, if you do not want to have the username be **SailPoint-user** use the "-c" flag and input your own syntax to create a unique user:

   SCIMInstaller.exe -c Client

   A user will be created called **Client-user**
4. Follow the prompts to enter the information outlined in the Pre-Install Checklist



Verify the Installation

1. Log into PrivateArk Client; locate and open the **SCIM Config** safe
2. Verify the presence of the following objects:

   Encryption-key - The SCIM Server uses a local cache to store objects retrieved from the Vault. Although no credentials (other than the ones in the SCIM Config safe, which are not stored on the cache) are retrieved, we encrypt the cache with this encryption key. The key is randomly generated, and not exposed, by the installer but can be changed if desired.

   > **NOTE:** If you have **encryptedBase64** enabled this key is also part of the encryption process of the PrivilegedData ID if the that is stored in the governance platform and presented to the SCIM for management purposes

   GlobalConfig.yml – This is the configuration file for the overall settings of the SCIM server. It is responsible for the setting of performance parameters and additional features that are added.

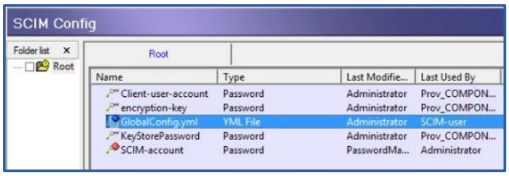
   SailPoint-account - This is a privileged account to allow the SailPoint IIQ authenticate its REST API requests to the SCIM Server. Because SailPoint doesn't yet use AIM, this account is not managed by the CPM, you'll have to store this password in IIQ when you define the CyberArk application. The password for this account must be the same as the SailPoint-user

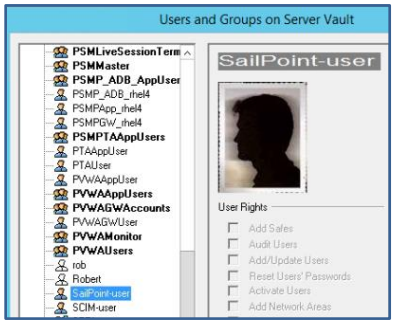
   SCIM-account - This is a privileged account, can be managed by the CPM, to allow the SCIM server to retrieve the password for SCIM-user thru an AIM Credential Provider call.

3. Verify that the following **Users** were created in the **PrivateArk Client:**
    a. Go to **Tools**; **Administrative Tools**
    b. Select **Users and Groups**
    c. Ensure the following users have been created:

SCIM-user - This is a CyberArk user with full privileges to create and manage Safes, Accounts, Permissions, and Users. This user is required by the PACLI interface on the SCIM server to login to the Vault and manage objects on behalf of client applications such as SailPoint.

SailPoint-user/Client-user - This is a CyberArk user for authenticating SailPoint requests made to the SCIM server using the REST API.





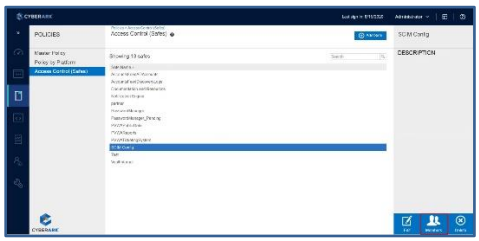
Post-Installation

SCIM Server Application ID

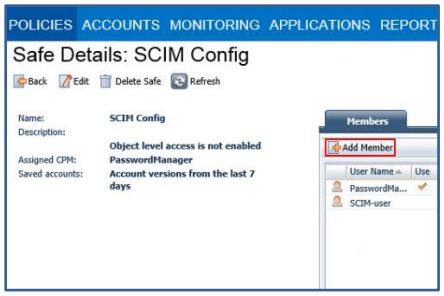
The SCIM Server uses the CyberArk-SCIM Application ID to access and retrieve objects inside the SCIM Config Safe in order to run CLI commands against the Vault. It is recommended to protect the Application ID with the Hash Authentication method. This will prevent any other executable to use the CyberArk-SCIM Application ID for authentication.

Adding the Provider User to the SCIM Config Safe

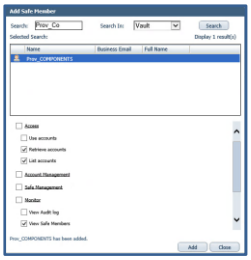
1. Click the **POLICIES** Tab
2. Click **Access Control (Safes)**; Select the **SCIM Config** safe; Click **Members**



3. In the Safe Details page Click **Add Member**



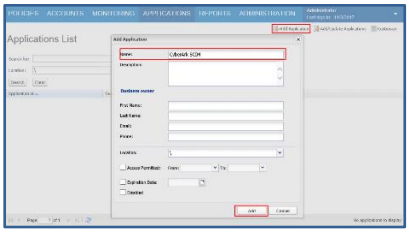
4. In the **Add Safe Member** window search for **Prov**
5. Select the Provider user that was created during the Credential Provider installation. (default username **Prov_<hostname>**)



6. Enable the **Retrieve Accounts, List Accounts,** and **View Safe Members** permission and Click **Add** then **Close**

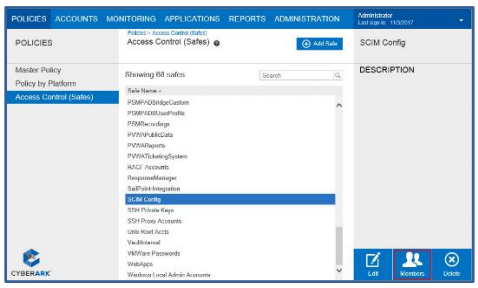
Creating the CyberArk-SCIM Application ID

1. Login to the PVWA with a user that has access to the **APPLICATIONS** tab
2. Click the **Applications** Tab; Select **Add Application**
3. In the Add Application window, enter **CyberArk-SCIM** in the Name field
4. Click **Add**

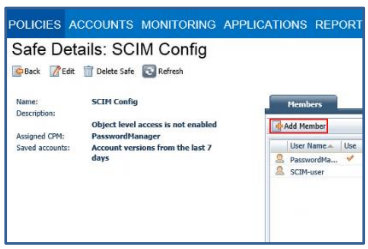


Add CyberArk-SCIM to the SCIM Config Safe

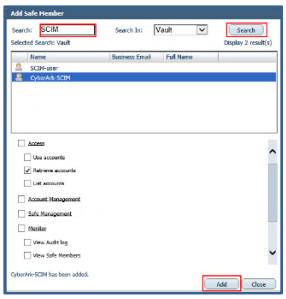
7. Click the **POLICIES** Tab
8. Click **Access Control (Safes)**; Select the **SCIM Config** safe; Click **Members**



9. In the Safe Details page Click **Add Member**



10. In the Add Safe Member window search for **SCIM**
11. Select **CyberArk-SCIM** and enable the **Retrieve Accounts** permission and Click **Add** then **Close**



Securing the Application ID

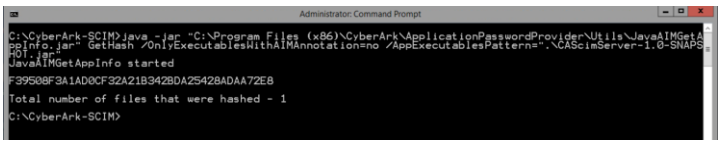
Generating a Hash Key

Hash Keys are the most secure way to protect a CyberArk application. The **JavaAIMGetAppInfo** utility is provided with the AIM Credential Provider that can be used to generate a Hash Key from a java executable.

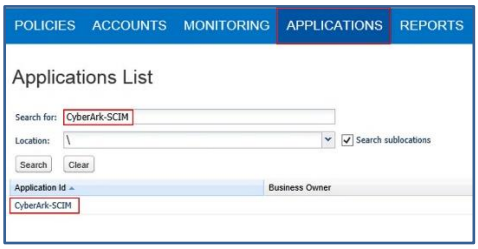
The Hash Key is dependent on the contents of the file it is being run against. Each time you update the SCIM Server the hash key for **CAScimServer-<VERSION>.jar** must be regenerated, and the steps below need to be repeated.

1. From the **CyberArk-SCIM** folder run the following command:

```
java -jar "C:\Program Files
(x86)\CyberArk\ApplicationPasswordProvider\Utils\JavaAIMGetAppInfo.ja
r" GetHash /OnlyExecutablesWithAIMAnnotation=yes
/AppExecutablesPattern=".\CAScimServer-<VERSION>.jar"
```



2. The output provided needs to be copied and placed as an **Authentication Method** in the PVWA
3. Login to the PVWA with a user that has access to the **APPLICATIONS** tab
4. Click the **Applications** Tab
5. Search for the **CyberArk-SCIM** Application ID and Click on the application



6. In the Application Details screen Click **Add**; then Select **Hash**

7. In the Add Hash Authentication window place the generated **Hash Key** from the JavaAIMGetAppInfo command and input it into the Hash field. Click **Add**

Testing the SCIM Server

There are multiple ways to test connectivity and functionality of the SCIM server, in this section we will outline the options available.

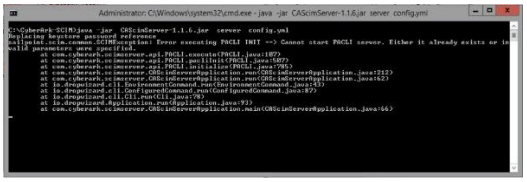
Starting the SCIM Service by command line

To start the SCIM Server from a command line, open a Windows command prompt as Administrator, CD **C:\CyberArk-SCIM**, then run:

```
java -jar CAScimServer-<VERSION>.jar server config.yml
```

If successful, this command will not produce any immediate output. If there is any output that brings you back to a command prompt, evaluate the errors and troubleshoot. As you use the SCIM Server, there will be output from this command. If you cannot troubleshoot the issue, please contact the support email provided at the end of this document for assistance.

NOTE: If you have created a Windows Service and the service is running, and you run this as well, it will create a PACLI error and will not function correctly. Only one can be run at a time.

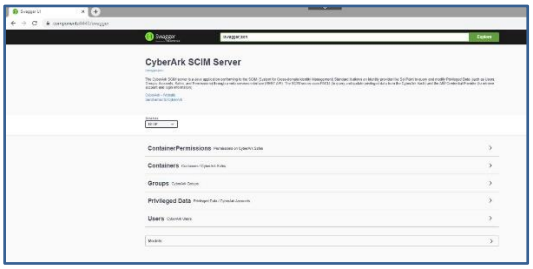


Accessing the Swagger UI

Open a web browser of your choice and navigate to the following address:

```
http://<address of SCIM Server>:8080/swagger
```

This will bring up an interactive Swagger UI to assist with running available commands. This will ensure everything is running correctly and permissions are configured correctly.



Sending REST API commands

The following table shows the list of parameters necessary to get a **Resource List**:

REST API Call

URL	`http://<SCIM IP>:8080/CyberArk/scim/v2/ResourceTypes`
HTTP Method	GET
Content Type	application/json
Authentication	• Basic Auth with **client-user** or **SailPoint-user** • Oauth with Bearer token

Using CURL

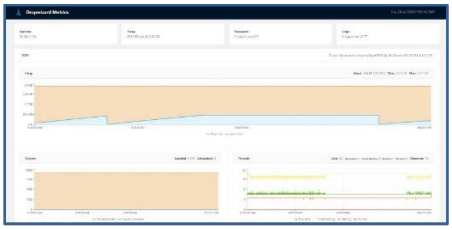
```
curl -u SailPoint-user:<PASSWORD> http://<SCIM SERVER
IP>:8080/CyberArk/scim/v2/ResourceTypes
```

Admin Console

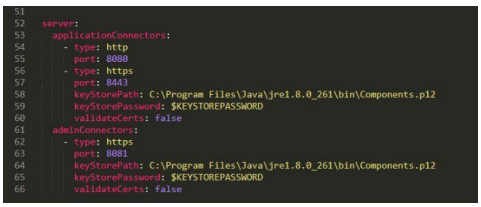
The Admin Console displays the activities of the SCIM and the JVM. This can be configured to use HTTPS in the config.yml just like the SCIM swagger API and can piggy-back off of the already existing java keystore and KEYSTOREPASSWORD dynamic reference. The address to access this feature is:

https://<SCIM_hostname>:<Port>/admin

Example:
https://cyberscim:8081/admin



This is the Admin Console configuration example from the **Config.yml**



```
server:
  applicationConnectors:
    - type: http
      port: 8080
    - type: https
      port: 8443
      keyStorePath: C:\Program Files\Java\jre1.8.0_261\bin\Components.p12
      keyStorePassword: $KEYSTOREPASSWORD
      validateCerts: false
  adminConnectors:
    - type: https
      port: 8081
      keyStorePath: C:\Program Files\Java\jre1.8.0_261\bin\Components.p12
      keyStorePassword: $KEYSTOREPASSWORD
      validateCerts: false
```

Configuring the SCIM Server for HTTPS

The SCIM server uses Dropwizard for the java keystore framework. Dropwizard follows the basic keytool commands for the management of certificates within the keystore.

Creating a PKCS12 keystore with a PFX/P12 certificate file

PKCS12 is the desired keystore type over JKS due to security of the keystore

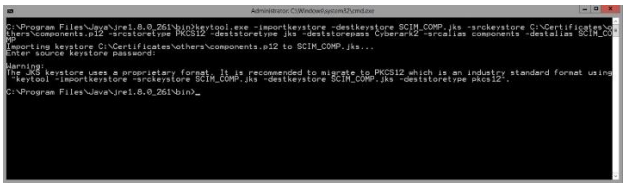
If you already have a PKCS12 certificate to install, here is the necessary syntax for importing the certificate as a jks keystore then converting it to a PKCS12.

Open a command window as Administrator and navigate to **C:\Program Files\Java\jre1.8.0.<version>\bin** and run the following commands:

```
keytool.exe -importkeystore -destkeystore <MY_KEYSTORE>.jks -
srckeystore <MY_FILE>.pfx/p12 -srcstoretype PKCS12 -deststoretype jks
-deststorepass <PASSWORD> -srcalias <ALIAS> -destalias <ALIAS_DEST>
```

The above command should prompt you for the password assigned to the certificate's key. Once you input the password you will get a warning suggesting converting it into a PKCS12:

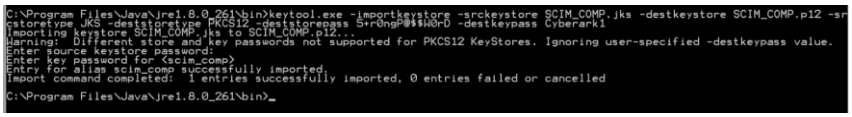
```
Warning:
The JKS keystore uses a proprietary format. It is recommended
to migrate to PKCS12 which is an industry standard format
using
 "keytool -importkeystore -srckeystore SCIM_COMP.jks -
destkeystore SCIM_COMP.jks -deststoretype pkcs12".
```



This is the command to run for converting the jks into a PKCS12. This is also another opportunity to set the password for the keystore that will eventually be set in the Config.yml

```
keytool.exe -importkeystore -srckeystore <MY_KEYSTORE>.jks -
destkeystore <MY_FILE>.p12 -srcstoretype JKS -deststoretype PKCS12 -
deststorepass <PASSWORD> -destkeypass <PASSWORD_of_original cert>
```

You will then be prompted for the password of the jks store that you set in the previous command, then prompted for the key password for the certificate. It is easier to have a single password for all entries and the security implications are minor as long as the final PKCS12 keystore password is complex.



Brief description of the variables in the above commands

Variable	Description
MY_FILE.p12	Path to the PKCS#12 file (.p12 or .pfx extension) that is going to be created
MY_KEYSTORE.jks	Path to the keystore that you want to convert
ALIAS	Name given to a certificate and key pairing, this is for the keytool API to easily identify the certificate and key pair stored in the keystore. The keytool API CANNOT discern between duplicate alias entries within the keystore. **This is also the friendly name or internal name of the certificate.**
ALIAS_DEST	Name that will match your certificate entry in the PKCS#12 file, you can keep the alias the same when creating the PKCS12 keystore and reference this keystore in the SCIM parameters file. "SCIM.foo.bar" for example
PASSWORD	Password for accessing the keystore

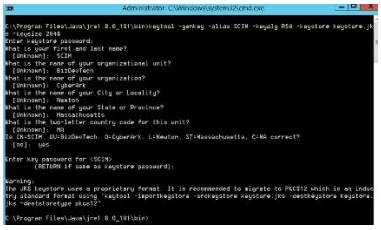
Creating a PKCS12 keystore from scratch

If you do not have a PKCS12 certificate ready to import, the following commands outline the generation of a keypair and then a CSR. Once the CSR gets signed and returned with the root CA cert (or intermediate), then you can create the keystore then convert it to a PKCS12 store.

First generate your keys

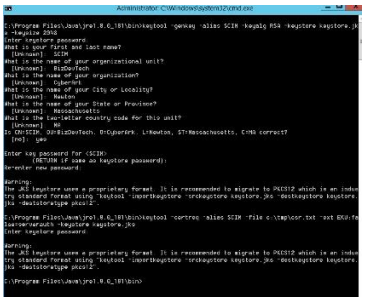
1. Open an Administrative cmd window and navigate to:
2. **C:\Program Files\Java\jre1.8.0.<version>\bin**
3. Run the following command:

```
keytool -genkey -alias <ALIAS> -keyalg RSA -keystore
<MY_KEYSTORE.jks> -keysize 2048
```



4. Then generate your CSR:

```
keytool -certreq -alias <ALIAS> -file csr.txt  -ext
EKU:false=serverauth -keystore <MY_KEYSTORE.jks>
```



5. Send the **csr.txt** to your certificate management team to have this request signed. They should provide a **signed certificate** and a **root CA** (or intermediate) certificate
6. You will then concatenate the contents of the **signed certificate** and **root CA cert**. Ensure the contents of the returned certificate are before the info from the root CA cert. Once you have created this certificate file, this is the file that will be imported into the keystore with the following command:

```
keytool -import -trustcacerts -keystore keystore.jks -storepass
<PASSWORD> -alias <ALIAS> -file all_combined.crt
```

7. Then convert the JKS keystore to a PKCS12 keystore:

```
keytool -importkeystore -srckeystore keystore.jks -destkeystore
<MY_FILE.p12> -srcstoretype JKS -deststoretype PKCS12 -destkeypass
<PASSWORD> -srcalias <ALIAS> -destalias <NEWER_ALIAS>
```

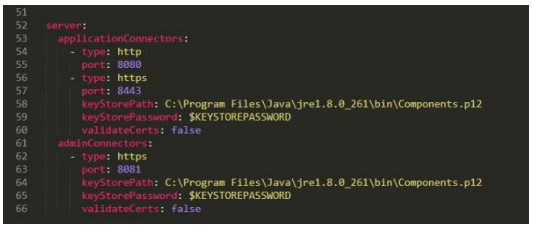
8. Import the root CA cert into the **trusted store** for the **Computer Account**
9. Modify your **SCIM config** file to reflect the created keystore with the instructions below

Configure the Config.yml

To configure the SCIM instance for SSL connectivity the settings need to be added to the configuration file.

Navigate to the **C:\Cyberark-SCIM** directory and edit the **Config.yml** and add the following parameters:

```
- type: https
  port: 8443
  keyStorePath: <Full Path here>\keystore.<keystore type>
  keyStoreType: <JKS or PKCS12>
  keyStorePassword: <PASSWORD>
  validateCerts: false
```



NOTE: The **validateCerts** parameter can be defined as:

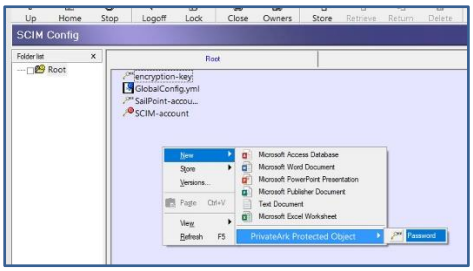
Whether or not to validate TLS certificates before starting. If enabled, Dropwizard will refuse to start with expired or otherwise invalid certificates. It is a convenience class that handles validation of certificates, aliases and keystores. It allows specifying Certificate Revocation List (CRL), as well as enabling CRL Distribution Points Protocol (CRLDP) certificate extension support, and also enabling On-Line Certificate Status Protocol (OCSP) support.

For this integration it is recommended to leave this parameter at **false**. Normal HTTPS setup should not need to have the server require a validation of its own certs, that can be managed at a different level. The reason we set it is because it is a mandatory parameter in the Config.yml. It can be set to true if you have one of the following correctly implemented Certificate Revocation List (CRL), CRL Distribution Points Protocol (CRLDP) enabled, certificate extension support, or On-Line Certificate Status Protocol (OCSP) support enabled, if at least one of these are not enabled and configured correctly the cert validation will fail unconditionally

Setting up a lookup variable for keystore password

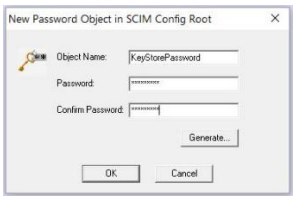
When setting up a keystore, the industry standard is to set the keystore password in clear text in your configuration file. This method is never going to be considered a **Best Practice** approach; so, we have developed the capability to Vault the keystore password and reference the stored credential as a variable in the config file allowing Dropwizard to access the credential using the SCIM-user account on the backend.

1. Once you have set up the keystore and you have the **password** for accessing the keystore, log in to the **PAClient** and navigate to the **SCIM Config** safe
2. Right-Click in the space available for the contents of the Safe
3. Hover over **New**; Select **PrivateArk Protected Object**
4. Click **Password**



5. In the New Password in SCIM Config Root window, enter **KeyStorePassword** for the **Object Name**.

   NOTE: This is case sensitive, ensure the syntax is exact



6. Enter and Confirm the **password** for the keystore
7. Click **OK**
8. On the SCIM Server navigate to **C:\Cyberark-SCIM** and edit the **Config.yml** file
9. For the **KeyStorePassword** parameter enter the following syntax exactly: **$KEYSTOREPASSWORD**

   NOTE: This is case sensitive, ensure the syntax is exact

Additional Processes

Setting up OAuth

OAuth is an open-standard authorization protocol or framework that describes how unrelated servers and services can safely allow authenticated access to their assets without actually sharing the initial, related, single logon credential. In authentication parlance, this is known as secure, third-party, user-agent, delegated authorization.

The CyberArk SCIM now allows for authentication to happen without the use of providing a username and password but relying on the use of a 3rd party authentication solution to provide a secure connection to the SCIM api. Scope is linked to a user by the SCIM Rights assigned to it in the authentication solution.

Example Syntax for GlobalConfig.yml

MSFT Azure example syntax

```
oauth2:
  jwksURI: "https://login.microsoftonline.com/<tenant-id>/discovery/v2.0/keys"
  issuer: "https://login.microsoftonline.com/<tenant-id>/v2.0"
  usernameField: "preferred_username"
```

Okta example syntax

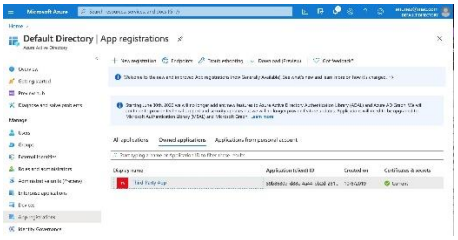
```
oauth2:
  jwksURI: "https://<tenant-uri>/oauth2/<API-auth-server>/v1/keys"
  issuer: "https://<tenant-uri>/oauth2/<API-auth-server>"
  usernameField: "sub"
```

Example syntax in the SCIMConfig.yml

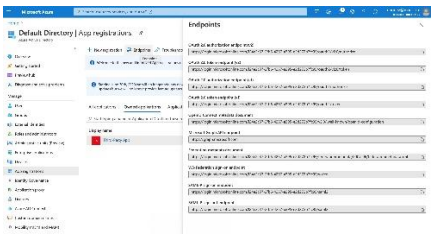
```
privilegedDataIDEncoding: encryptedBase64

oauth2:
  jwksURI: "https://login.microsoftonline.com/f45ca6bb-2fb6/discovery/v2.0/keys"
  issuer: "https://login.microsoftonline.com/f45ca6bb-2fb6/v2.0"
  usernameField: "preferred_username"

ignoreSafes:
  -System
  -passwordmanager

=====================================

privilegedDataIDEncoding: encryptedBase64

oauth2:
  jwksURI: "https://dev-228923.cyberark.com/oauth2/scim-auth/v1/keys"
  issuer: "https://dev-228923.cyberark.com/oauth2/scim-auth"
  usernameField: "sub"

ignoreSafes:
  -System
  -passwordmanager
```

Configuring OAuth with MSFT Azure AD

1. Authenticate to the Azure portal and navigate to **Azure AD**
2. In your **Directory** page, select **App registrations**
3. From here you will need to register the SCIM server as an application



4. In the **App registrations** page ***NOT*** the newly registered app page, click **Endpoints** at the top of the page



5. In the Endpoints page locate the **OpenID Connect Metadata Document** field and copy the URI
6. Paste the uri into the browser and it will return a json dictionary of metadata
7. Find and copy the values for the following keys:
   - **jwksURI**
   - **issuer**
   - **usernameField** is going to be one of the Claims in the JWT token. In our development process we have seen the username come in under the **preferred_username** value
8. These are going to be the values that need to be entered into the **GlobalConfig.yml**

Defining Scope

Scope is the permissions set given to the user that will be used to authenticate to the SCIM server from the governance platform. Reference the SCIM Rights section to properly assign scope for enacting the action needed for the user being authenticated. The scope will be a claim in the JWT that identifies to the SCIM Server the permissions for the user that has been authenticated.

Once back at the **Directory page**; Select the registered app

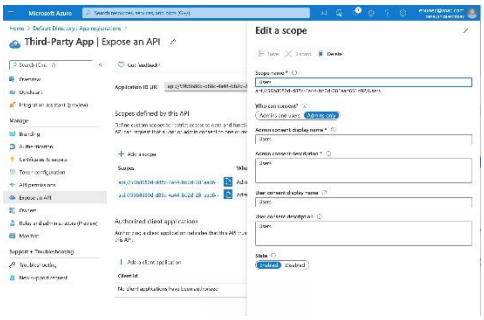
Once in the **Application page**; Select **Expose an API**

In the **Expose an API** page; Select **Add a Scope**

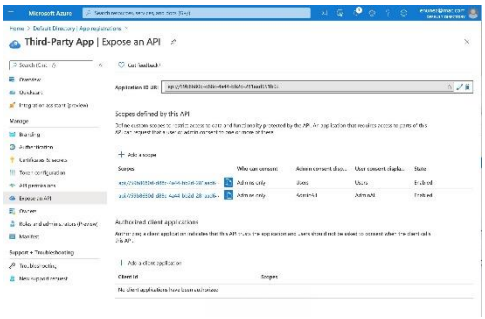
If this is the first time adding a scope you will be prompted to define and **Application ID URI**, either provide a value or confirm the default

In the **Add a Scope** section, the **Scope name** needs to match the value outlined in the SCIM Rights section.

The example below illustrates read-only access to the **Users** grouping of the SCIM api



Once you have filled out all of the necessary fields and click **Save** you will see the **Scopes** reflected in the **Expose an API** page



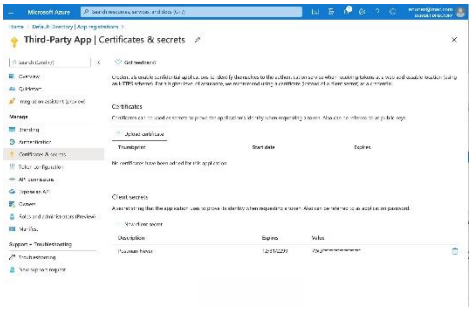
Certificates and Secrets

From here you will need to set up the Authentication portion that is in line with the governance platform's specification

In order for any third-party that needs to authenticate to the SCIM Server it will first need to "negotiate" an access token with the Identity Vendor (Azure AD)

The SCIM Server makes sure that when it receives an access code during the authentication process, the Identity Vendor can be validated. This is done by providing the governance platform a client id and secret from the Identity Vendor

In the Expose an API page there is a sub-section labeled **Certificates and secrets**, this is where these values can be generated.

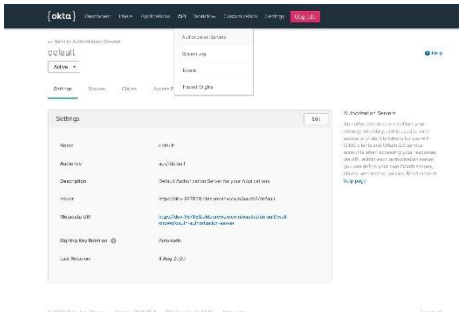


Configuring OAuth with Okta

In the Okta administrative view select the **API** tab and navigate to **Authorization Servers**

Choose the Authorization Server that will authenticate the identities accessing the SCIM server

In the **Metadata URI** field click the **URI**



In the page that loads, find and copy the values for the following keys

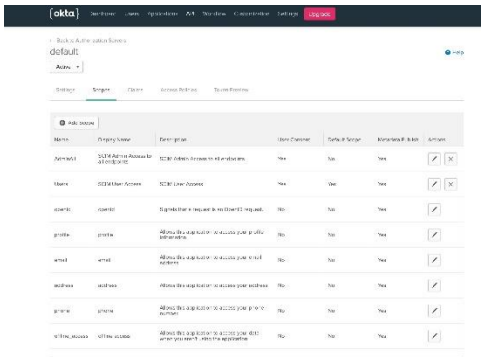
- **jwksURI**
- **issuer**
- **usernameField** is going to be one of the Claims in the JWT token. In our development process we have seen the username come in under the "**sub"** claim

These are going to be the values that need to be entered into the **GlobalConfig.yml**

Defining Scope

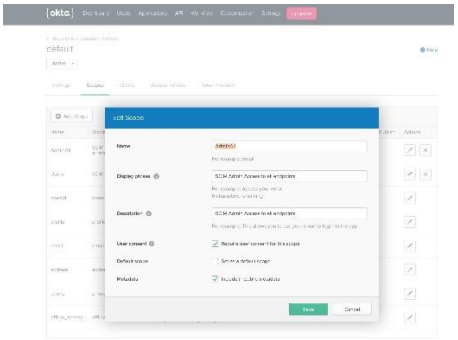
Scope is the permissions set given to the user that will be used to authenticate to the SCIM server from the governance platform. Reference the SCIM Rights section to properly assign scope for enacting the action needed for the user being authenticated. The scope will be a claim in the JWT that identifies to the SCIM Server the permissions for the user that has been authenticated.

In the Authentication Server page that you selected, click the **Scopes** tab



Click **Add Scope**

In the **Edit Scope** page, the **Name** field needs to match the value outlined in the SCIM Rights section



Click **Save** and see the new scope reflected in the **Scopes** page.

Tokens and Trusted Origins

From here you will need to set up the Authentication portion that is in line with the governance platform's specification

In order for any third-party that needs to authenticate to the SCIM Server it will first need to "negotiate" an access token with the Identity Vendor (Okta)

The SCIM Server makes sure that when it receives an access code during the authentication process, the Identity Vendor can be validated. This is done by providing the governance platform a client id and secret from the Identity Vendor

In the API page there are tabs to manage **Tokens** and **Trusted Origins**, this is where these values can be generated.

Setting up the Config YAML files

PrivilegedData IDs

To identify account objects in the Vault the SCIM server gives each account object (Privileged Data) a PrivilegedData ID which consists of the following syntax to identify the object:

Safe:::Folder:::Object Name

Example:

Windows Local Account:::Root:::Operating System-WindowsDesktopLocalAccountsRotationalPolicy-10.0.0.10-x_accountA

The SCIM server will run this value through a Hex table to make a Hex encoded value of this string. This process can lead to the end value of this string being over the 450-character limit in the SailPoint system and cause Aggregation tasks to fail and not recover.

The newly added **privilegedDataIDEncoding** parameter allows admins to use one of three available methods:

<null> / default: Uses the original Hex encoding of PrivilegedData ID

base64: This will take the clear text ID syntax and base64 encode that value and transmit the base64 encoded value as the new ID

encryptedBase64: This will encrypt the clear text value with the encryption key that is stored in the SCIM Config safe for encrypting the cache. Then it will take the encrypted value and base64 encode it.

NOTE: If the encryption key is ever rotated, then the PrivilegedData IDs will change. A new aggregation task will need to be run as the data stored in the Governance platform will have been tied to the old PrivilegedData IDs.

GlobalConfig.yml stored in SCIM Config safe

This file contains a list of **Safes**, **Users**, and **Groups** to be ignored by the SCIM sever. The version provided with the distribution is just a sample, you may add and remove elements from these lists as you see fit. The file is uploaded to the **SCIM Config** safe by the installer. **To modify this file, you most retrieve it from the safe, modify it, and save it back**. To have the changes reflected in the SCIM server the SCIM service and the **CyberArk Application Password Provider** service need to be restarted.

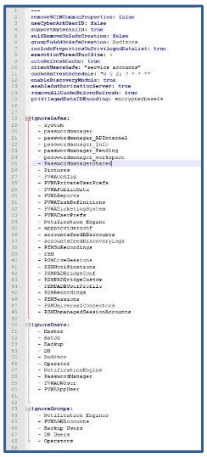
GlobalConfig.yml

Parameter	Description
ignoreSafes	TreeSet<String>: - newTreeSet<String> Case Insensitive List of Safes to ignore
ignoreUsers	TreeSet<String>: - newTreeSet<String> Case Insensitive List of Users to ignore
ignoreGroups	TreeSet<String>: - newTreeSet<String> Case Insensitive List of Groups to ignore
removeSCIMCommonProperties	Boolean - Whether to identify id and externalid as a “common attribute”. If set to true, when querying the Schemas endpoint these properties will be stripped from the result. https://tools.ietf.org/html/rfc7643#section-3.1
useCyberArkUserID	Boolean - Whether to use CyberArk's internal user id (instead of username)
supportExternalID	Boolean - Whether SCIM will support external ids. Some governance platforms have their own ids and if set to true SCIM will create a mapping file called Users-ExternalIDs.yml and store it in the SCIM Config safe.
selfRemoveOnSafeCreation	Boolean - If true, the "SCIM-user" will be removed automatically after a safe is provisioned via SCIM, and after adding "SCIM-user" to the group defined in groupToAddOnSafeCreation
groupToAddOnSafeCreation	String - If set, it will add a single group to the newly provisioned safe defined in this parameter in order for the scim-user

	to continue to have access to the safe provisioned when selfRemoveOnSafeCreation is set to true
includePropertiesOnPrivilegedDataList	Boolean - If true, when querying the PrivilegedData endpoint, it will also return the file categories (properties) associated to the account object on the resulting list
executionThreadPoolSize	Integer - If > 0 it will execute in the multithreading pool with size defined. The suggestion is to set this to the number of cores assigned to the SCIM Server machine.
autoRefreshCache	Boolean - If true, it will execute auto-refresh of the cache objects by endpoint groupings when the cacheRefresh parameter elapses, it checks if the creation date has expired every 60 seconds
clientUsersSafe	String - Allowing for admins to have the ability to store the account object for the governance solution (client-user or SailPoint-user) in a safe other than the SCIM Config safe
cacheRefreshSchedule	String - Cron expression, encased in quotations to enable scheduling of a cache refresh autoRefreshcache must be enabled for this to function. Can also be set in local Config.yml
enableDiscoveryModule	Boolean - Enables the CyberArk Discovery Module and allows the reading of the DiscoveryConfig.yml file
removeAllCacheBeforeRefresh	Boolean – indicates that when a cache refresh is enacted, the entire cache will be deleted (cacheRefresh removes groupings that have expired) Can also be set in local Config.yml
privilegedDataIDEncoding	String (base64, encryptedbase64) – **base64**: Enables the SCIM to base64 encode the PrivilegedData IDs being sent to SailPoint **encryptedBase64**: Allows for the PrivilegedData IDs to be encrypted, **then** Base64 encoding of the encrypted value. **<null> <default>**: Just a general Hex salting of the PrivilegedData IDs that is the OOB behavior.
internalRefreshPageSize	Integer – Number of entries per page (default 200)

batchesPerInitPACLI	Integer – Number of batches to be processed before SCIM will completely close the PACLI connection to the Vault and restart PACLI. If this parameter is not set it will not restart. Use this parameter ONLY when necessary.

header	key	value All values for this dictionary are encased in quotations
oauth2:		
	jwksURI	String – URI value for the public keysets for signature validation
	issuer	String- URI for the entity that generates the auth token
	usernameField (optional)	String – default value “sub” the claim in the JWT returned from the authenticating solution to identify the user being authenticated
	signatureAlgorithm (optional)	String – default value “RS256” The encryption algorithm used to decrypt the incoming auth JWTs



Pages and Batches

In large environments during creation of cache, it was observed that results from issued PACLI commands to the Vault returned large chunks of data (ie USERSLIST). When SCIM would go to parse through this data we saw that failures would happen as it would parse from memory. To combat these issues we implemented pagination and we have exposed a parameter to set the number of entries per page. If you set the **internalRefreshPageSize** to 1000 and you have 100,000 entiries it will break the entries into 100 different pages. There is also a limitation on the capability that requires the logic to start from the first entry and go all the way through each page

until it reaches the page it is concerned with. Back to our scenario, for the first page it will go from 1-1000 and parse that information into cache items, but to get to the 2nd page it will parse from 1-1000 so it can get to 1001 and so on for 100 pages.

Batches is described as the number of entries parsed on a page calculated by the number of threads set in the executionThreadPoolSize times 2. If the number of threads is 4 then a batch is 8 and it will parse 8 entries into cache until the page is completed. In our scenario we have 1000 entries per page and now we are parsing 8 entries per batch, it will be 125 batches to parse through a page. We have seen in certain environments that PACLI will begin to lose connection to the Vault and the syntax "**Unable to communicate PACLI server. (Including Error Code: 1)... trying again**" will begin to populate the CAScimServer.log file. The **batchesPerInitPACLI** parameter was introduced allowing the admin to choose how many batches are parsed before a complete restart of PACLI will be conducted to ensure PACLI has a fresh session.

Config.yml stored in CyberArk-SCIM folder (CyberArk TreeSet)

This configuration file is for configuring the "connection" parameters for the SCIM server and is comprised of three TreeSets, **logging**, **cyberark**, and **server**. Here we describe a list of the parameters on the file you're allowed to modify. Modifying parameters not described here is possible but will not be supported.

In the **server** TreeSet you can modify the ports for http, https, and adminConnectors. This is also the area for setting your keystore parameters for SSL connectivity. These settings are outlined in the **Configure the Config.yml** section of this guide.

CyberArk TreeSet

Parameter	Description
parmFile	String - vault.ini to use
cacheRefresh	Integer - how long, in seconds, how often the privileged data from the Vault is refreshed. Certain events, such as creating a user or group, can invalidate the cache causing it to be refreshed sooner. The default is 600 seconds
safe	String – Safe containing the SCIM account object. By default, this is the SCIM Config Safe
folder	String – Folder containing the SCIM account object. By default, this is root
objectName	String – Unique object name of the SCIM account. By default, this is SCIM-account
appID	String – Application ID created in SCIM Server Application ID section
usePackagedPACLI	Boolean - Whether to use the PACLI that comes embedded with the SCIM server distribution.
debugPACLI	Boolean – default is true

	If set to false it will not print the PACLI commands to the logs. This will significantly decrease the size of the cascimserver log file
pacliSessionID	String - Session ID to use for PACLI statements other than the default 0
retriesForAuthErrors	Integer – Number of authentication retires when SCIM encounters Authentication Error (ITATS004E)
cacheRefreshSchedule	String - Cron expression, encased in quotations to enable scheduling of a cache refresh autoRefreshcache must be enabled for this to function. Overrides same setting in GlobalConfig.yml
removeAllCacheBeforeRefresh	Boolean – indicates that when a cache refresh is enacted, the entire cache will be deleted (cacheRefresh removes groupings that have expired) Overrides same setting in GlobalConfig.yml

```
cyberark:
parmFile: Vault.ini
safe: SCIM Config
folder: root
objectName: SCIM-account
cacheRefresh: 36000
appID: CyberArk-SCIM
debugPACLI: false
#cacheRefreshSchedule: "0 0 21 ? * * *"
#removeAllCacheBeforeRefresh: true
```

Enable Logging

The logging functionality is built using the Dropwizard framework which uses **Logback** for its logging backend. It provides a Simple Logging Facade for Java (SLF4J) implementation allowing for granular customization to achieve the desired logging output.

This section will outline the basic settings and expectations of output for the various syntax.

Config.yml

In **C:\Cyberark-SCIM** directory the **Config.yml** file is where to configure the logging output:

Config.yml TreeSet definitions

TreeSet	Description
logging	TreeSet to indicate parent logging settings
level	The default level of all loggers in this branch. INFO, DEBUG, TRACE, ERROR **NOTE**: multiple entries can be set and are comma delimited. (i.e. INFO,ERROR)
appenders	TreeSet to define the destination of the logging information
loggers	Header to define logger-specific settings
additive	This boolean parameter stops/allows the current logger (or anything under it) from using the root logger

Config.yml file logging settings

Setting	Description
type	Parameter to indicate the output type
currentLogFilename	The file to which current statements will be logged
archivedLogFilenamePattern	When the log file rotates, the archived log will be renamed to this. The %d is replaced with the previous day in (yyyy-MM-dd) format
archivedFileCount	default is 5 The number of archived files to keep
logFormat	Pattern of the line of text in the output. To add milliseconds, append 'SSS' to the date-time stamp
includeCallerData	Boolean parameter to allow data to be dynamically called in the logFormat.
maxFileSize	default is unlimited The maximum size of the currently active file before a rollover is triggered

These 4 logs that are configured in the config.yml that comes with the zip file from MarketPlace

Logging – This is the parent log that will output actions within the core framework.

com.cyberark – This is an output for anything related to available SCIM parameters, or actions outside of the core framework.

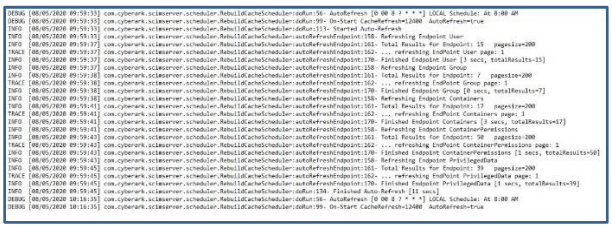
scimserver.audit – This only shows the actions from external actions (authentication, commands passed, queries)

Optional additions:

com.cyberark.discovery – This is an output of the work being done by the CyberArk Discovery Module

service.log – This is an output of activities conducted by NSSM, most notably, this will print out the HTTP status code for all incoming requests and will allow for cross-referencing of 500 HTTP status codes. If there is a failure on the IGA side and the request is fulfilled with a 200 status code, we know that the failure resides outside of the SCIM server.

com.cyberark.scimserver.scheduler – This is an output of the scheduling of the cache rebuild. This will outline the number of items for each group and how long it took for the cache to build. This will assist in identifying how many objects are supposed to be in cache when the cache refresh has been completed.



DropWizard Reference Material

https://www.dropwizard.io/en/latest/manual/core.html#logging

https://logback.qos.ch/manual/layouts.html#conversionWord

https://www.dropwizard.io/en/latest/manual/configuration.html

Cache Refresh Parameters

There are 5 'groupings' that make up the CyberArk SCIM cache to align with the api endpoints:

- Users
- Groups
- Containers (Safe)
- ContainerPermissions (Safe permissions)
- PrivilegedData (Account Objects)

The CyberArk SCIM caching parameters were introduced in phases. The first parameter was the **cacheRefresh** parameter and it is a bit of a misnomer because this parameter doesn't actually "refresh" anything it is just a cache 'validity' parameter. What it does is invalidate the data in cache based on this value (in seconds).

There are 2 mechanisms that will cause the cache to go and rebuild.

1. An incoming API call to the CyberArk SCIM will force the logic to check if the data is valid based on the **cacheRefresh** value. If it is invalid when the request comes in, then the logic will go out and rebuild that data.
2. The other mechanism is the **cacheRefreshSchedule** and it is controlled using a cron string to schedule the SCIM logic to check if cache is valid based on the

cacheRefresh value. If the cache is valid when the check is performed, then it does nothing. If the cache is invalid, it will trigger a cache rebuild for the grouping that is invalid.

- The **removeAllCacheBeforeRefresh** parameter is a mechanism that invalidates all the cache prior to the **cacheRefreshSchedule** evalution. Therefore forcing the logic to go and build the entire cache during each scheduled check.
- The **cacheRefreshSchedule** and **removeAllCacheBeforeRefresh** parameters can be set in EITHER (not both) the **GlobalConfig.yml** in the SCIM Config safe to allow for this schedule to be enacted for the entire architecture or in the **Config.yml** of an individual CyberArk-SCIM folder so the task only exists for that specific SCIM server.
- The **autoCacheRefresh** needs to be set to true in the "**GlobalConfig.yml**" for the scheduler to work

Examples

To schedule a validation to occur at **3:30am** everyday on a single CyberArk SCIM server:

"**GlobalConfig.yml**" stored in SCIM Config safe in the PrivateArk Client

```
autoRefreshCache: true
```

"**Config.yml**" stored in CyberArk-SCIM folder of individual SCIM server

```
cacheRefreshSchedule: "0 30 3 ? * * *"
removeAllCacheBeforeRefresh: true
```

For more information on the creating the cron expression above visit:

https://www.freeformatter.com/cron-expression-generator-quartz.html

Example of schedule cron job to occur at **3:30am** everyday on all CyberArk SCIM servers:

"**GlobalConfig.yml**" stored in SCIM Config safe in the PrivateArk Client

```
autoRefreshCache: true
cacheRefreshSchedule: "0 30 3 ? * * *"
removeAllCacheBeforeRefresh: true
```

When starting the SCIM service, you can open the **CAScimScheduler.log** file, on the first line, to get a plain text output of what times the scheduler is set to validate cache.

Decreasing the **cacheRefreshSchedule** or enabling the **removeAllCacheBeforeRefresh** does not hinder the performance, it just provides a mechanism to assist in managing the cache. Without the **cacheRefreshSchedule** parameter, incoming requests to the CyberArk SCIM API will be the only mechanism that forces the logic to build cache and that can have a severely adverse effect on performance as the request will have to wait for the data to populate before the request can be fulfilled.

Installing a Windows SCIM Service

Java Heap Optimization

The SCIM is written in Java and it requires JVM to run, meaning that you do not need JDK to be installed, JRE will suffice. It is suggested to have the JRE path in a **%JAVA_HOME%** Environment Variable.

The most recent version of the SCIM allows for more performance management of the JVM with a string emplaced in the **InstallService.bat** file, example below:

Line14 of the above example shows the string that will be placed in the service that runs the "CyberArk SCIM Server" service. The values of note here are the **-Xms8g -Xmx8g** values that indicate the amount of memory allocated to the JVM. The Xms value is for initial and minimum heap size and the Xmx is the maximum heap size for JVM.

The suggested value for these parameters is 8/10 of the RAM allocated to the server, so if there is 10GB of RAM allocated, then 8GB should be allocated to the JVM reflected just like the example shows. If the server has 32GB allocated, then the suggested values should look like **-Xms28g -Xmx28g**.

Java 64-bit is required for this process to work

Click here for more JVM tuning parameters and guidelines.

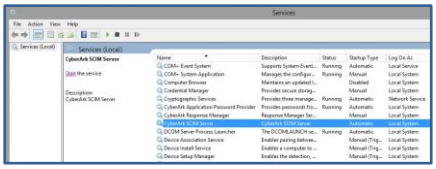
Running the batch scripts

Located in the CyberArk-SCIM directory there are two batch scripts provided for installing and uninstalling the SCIM Server as a windows Service:

InstallService.bat

RemoveService.bat

1. Open a command window as Administrator
2. Navigate to **C:\Cyberark-SCIM**
3. Run **InstallService.bat** script
4. Once complete you'll be able to see the **CyberArk SCIM Server** in the Services GUI:



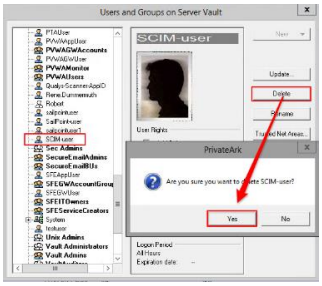
From this point you'll be able to manage it (start, stop, and restart) like any other windows service.

To remove the Service, follow the instructions above to run the **RemoveService.bat**

Uninstall Procedure

Being that the SCIM installer script can only be ran once, in order to run it a second time you must **uninstall** all the SCIM objects first:

1. Log into the **PrivateArk client** and make the following modifications:
2. **Delete** the **SCIM Config** safe by opening it but **do not step in**, then Right-Click and **delete** it.
   NOTE: Due to retention policies you might not be able to delete the safe immediately. If that's the case, **rename** it and wait until the end of the retention period to delete it.
3. Go to **Tools**; **Administrative Tools** and Select **Users and Groups**
4. Locate the **SCIM-user** and Click delete
5. Do the same for the **SailPoint-user**, if you created it before.



Upgrading the SCIM Server

Each time you download a new version of the **CyberArk SCIM** Server from the **Support Vault** you need to execute the following steps:

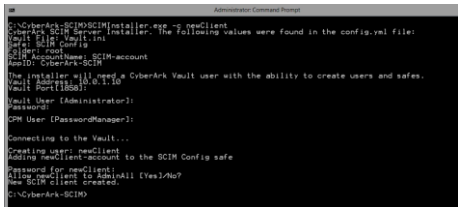
1. Stop the SCIM Server service.
2. Replace the old CAScimServer<VERSION>.jar with the file in the new from the distribution
   NOTE: either delete the old Jar file or add the extension .OLD
3. If you run the SCIM Server as a **Windows service**.
   a. Run the **RemoveService.bat** script
   b. Run the **InstallService.bat** script
4. **If you're using a Hash Key, regenerate it** for the new CAScimServer<VERSION>.jar file using the JavaAIMGetAppInfo utility Instructions for this are outlined in the **Generating a Hash Key** section of this guide
5. **Start** the SCIM Server service.

Creating additional SCIM Clients

Any client making a REST API call to the SCIM Server needs to be authorized and given the appropriate set of permissions (See **SCIM Server permissions** section). The installer allows you to define a client for SailPoint, but you may also use the installer to create additional clients for other applications. The client authenticates using **Basic Authentication** (username/password) which are stored in a privileged

account inside the Vault and we **strongly recommend the client application uses AIM to retrieve this credential.**

The Installer allows you to add a new SCIM client with the **–c** option:



The installer will create a new user in the Vault with the name specified by the **–c** option. This is the user that will be used to authenticate to the SCIM server from your application. It will also create an account in the Vault with the purpose of managing the credential for the new user. However automatic management for the account is initially disabled:

If your SCIM client application already integrates with the CyberArk Vault and is able to retrieve the credential using the CyberArk Application Identity Manager (AIM) component, you can **uncheck** the "**Disable automatic management for this account**" box and allow CyberArk to rotate the credential. This also avoids having to save the credential within your client application. Otherwise you'll have to store the username and password you just created within the client application.

SCIM Rights server permissions

This is the permission set that is applied to the Scope (scp) claim in the Oauth authentication flow.

The following permissions set is available to be applied to the user authenticating from the governance platform and is applied at the SCIM server level:

***Denotes a read-only permission**

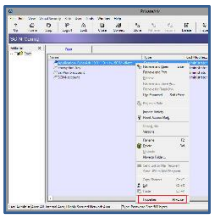
```
AdminAll
  AdminUsers
    *Users
  AdminGroups
    *Groups
  AdminContainers
    *Containers
  AdminPrivilegedData
    *PrivilegedData
  AdminACL
    *ACL
```

Permissions prefixed with **Admin** give the right to both query and alter the content of the objects they refer to. For example, **AdminUsers** allow a SCIM Server client to use the REST API to Query, Create, Update, and Delete users.

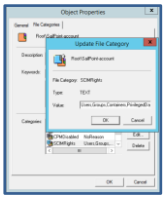
Permissions without the **Admin** prefix give the right only to query the object they're referring to. For example, **Users** allow a SCIM Server client to use the REST API to Query users, nothing more.

By the above you can infer that **AdminUsers** implies **Users**, therefore you only need to specify the first one to allow a SCIM Server client full control over users. By the same token **AdminAll** contains **Admin** permissions on all endpoints. The SCIM Server installer will not ask for a lesser privilege once a higher one has been granted. For example, if you grant **AdminUsers** to a client identity, it will not ask for **Users**. Similarly, if you grant **AdminAll**, it will not ask for any other permission.

If you need to modify the permissions an existing SCIM Client has, this can be done by modifying the associated account. Each SCIM Client created with the installer has both a CyberArk User and a Privileged Account associated to that CyberArk User in order to manage the user credential and to allow you to retrieve that credential using AIM. There is a **SCIMRights File Category** associated to the SCIM Client account which contains a list of permissions of what the SCIM Client is allowed to do. You can modify this to change its permissions.
Login to the PrivateArk Client and open (double click) the **SCIM Config** safe. The new SCIM client account created on the previous step should be there. Right-Click on it and Select **Properties**:



On the **Object Properties** dialog Click on the **File Categories** tab then scroll down and Double Click on the **SCIMRights** File Category:



Then type in the list of comma separated **SCIMRights** permissions you want the SCIM Client to have.

The permissionsMap.yml file

The permissionsMap yaml file is a configuration file allowing for the grouping of permissions applied to CyberArk users. It allows you to group a set of CyberArk Permissions/Rights/Roles and send the name of that group to the client (instead of all the individual Permissions/Rights/Roles) in order to simplify management by reducing the level of granularity you see on the SCIM Client. For example:

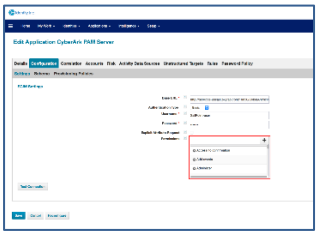
```
- name: Approver
  includesPermissions:
    - List
    - Supervise
    - View audit
```

```
- View permissions
```

The list of permissions (**List**, **Supervise**, **View audit**, and **View permissions**) will be aggregated into **Approver** before sending it to the client, so the client will only see **Approver**. The file contains a suggested list of mappings, you should modify this file as you see fit.

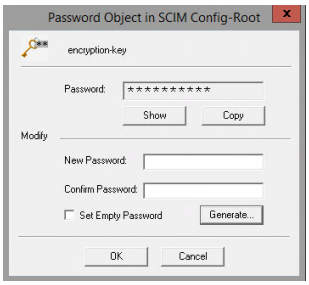
The file is shipped with the distribution but not stored in the **SCIM Config** safe by the installer. After you customize the **permissionsMap.yml.SAMPLE** you must rename it to **permissionsMap.yml**, store it in the **SCIM Config** safe and restart the SCIM server for the mapping to take effect.

If you're using the SCIM server with IdentityIQ edit the **CyberArk PAM Server** application configuration to list all the mappings you want IdentityIQ to have access to:



Changing the Cache Encryption Key

If you desire to change the cache encryption key, first you'll need to stop the SCIM Server and delete all files in the cache folder. Then login to the Vault using the PrivateArk Client, open the **SCIM Config** safe and Double Click on the **encryption-key** object:



Here you can either **Generate**, or manually enter a new key. Then Click **OK** to save. A restart of the SCIM Server will be necessary and if you are encrypting PrivilegedData IDs and new IDs will be assigned to the Account object and a **new aggregation will be required** to remap the IDs stored in SailPoint to the new value assigned to account objects by SCIM.

SailPoint IdentityIQ Implementation

Defining the CyberArk Application

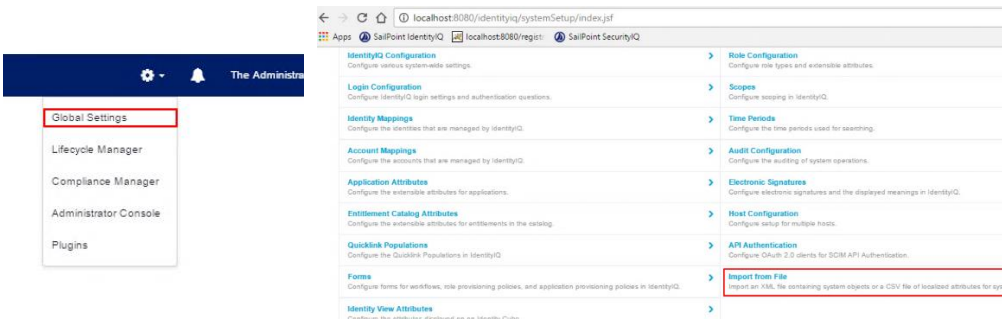
The Distribution file contains a folder named **SailPoint-Objects** with the following xml files inside:

Name	Date modified	Type	Size
AccountAggregation-Task	11/7/2017 8:51 AM	XML File	2 KB
AccountGroupAggregation-Task	11/7/2017 8:51 AM	XML File	2 KB
Application	11/7/2017 9:40 AM	XML File	41 KB
PAM-Tasks	11/7/2017 9:15 AM	XML File	2 KB
TargetCollector	11/7/2017 9:42 AM	XML File	9 KB

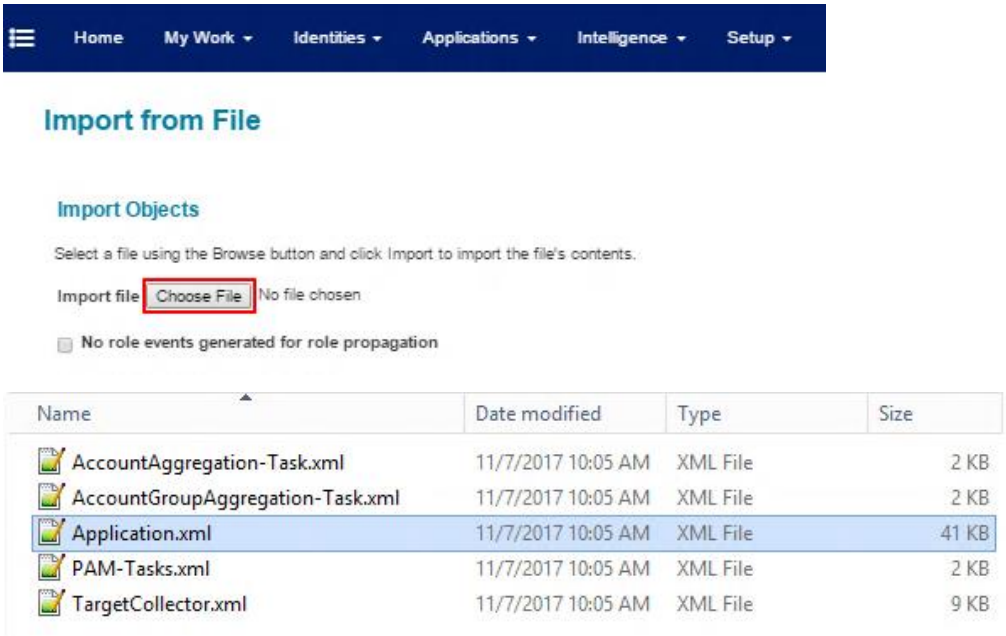
You'll need to load the files into IIQ in the following order:

Application.xml
AccountAggregation-Task.xml
AccountGroupAggregation-Task.xml
TargetCollector.xml
PAM-Tasks.xml

1. Login to SailPoint IIQ and Click the Settings icon in the top banner and Select **Global Settings**
2. In the Global Settings window Select **Import from File**:



3. In the Import Objects section Click on **Choose File** and Select the **Application.xml**:

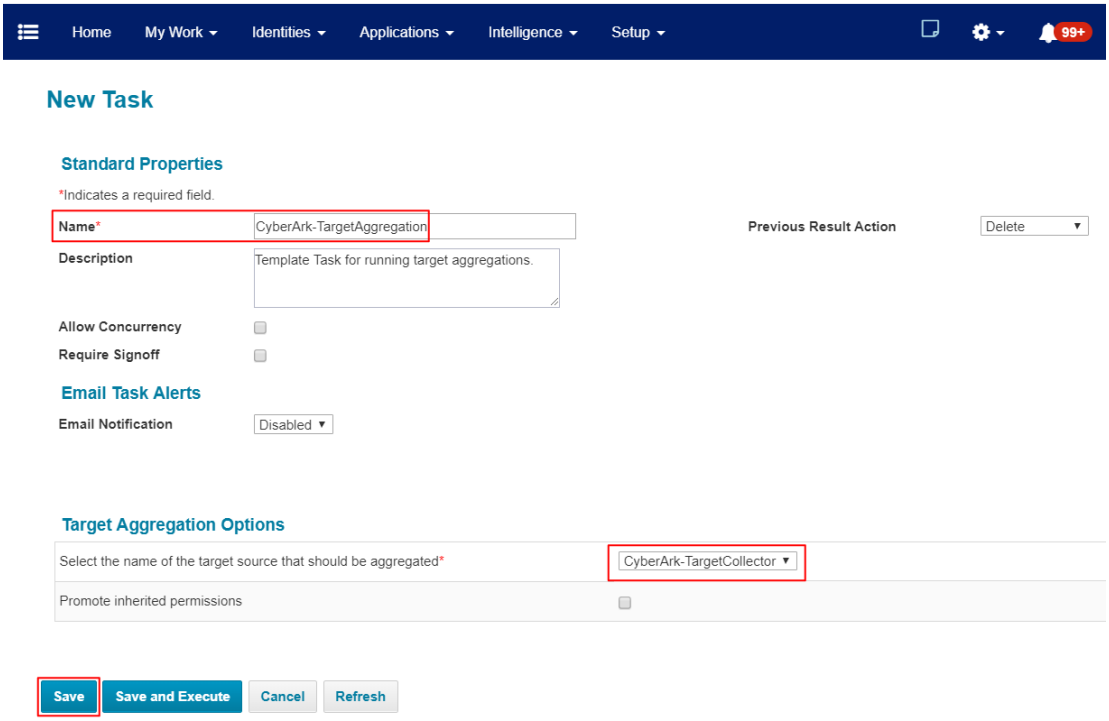


4. Click **Import**:

5. Repeat steps 3 and 4 for the remaining files in the order specified above.

Create the CyberArk-TargetAggregation Task

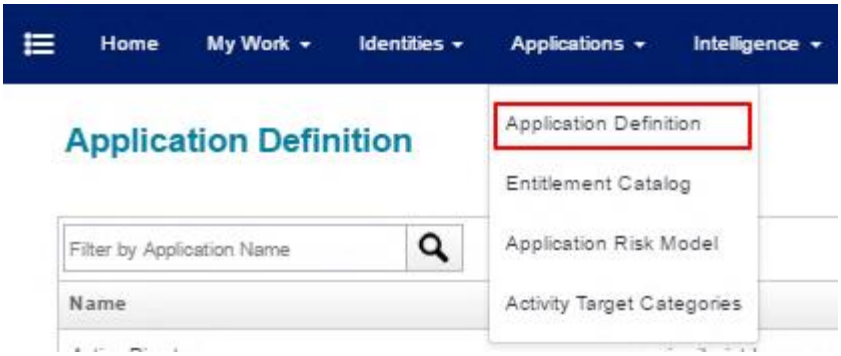
1. Click **Setup;** Select **Tasks**
2. Click on **New Task** and Select **Target Aggregation** from the list.
3. On the New Task screen enter **CyberArk-TargetAggregation** in the Name field:



4. Click on **Save**.

Edit the CyberArk PAM Server application definition

1. In the Application Definition page Click the **Applications** Tab
2. Select **Application Definition**:

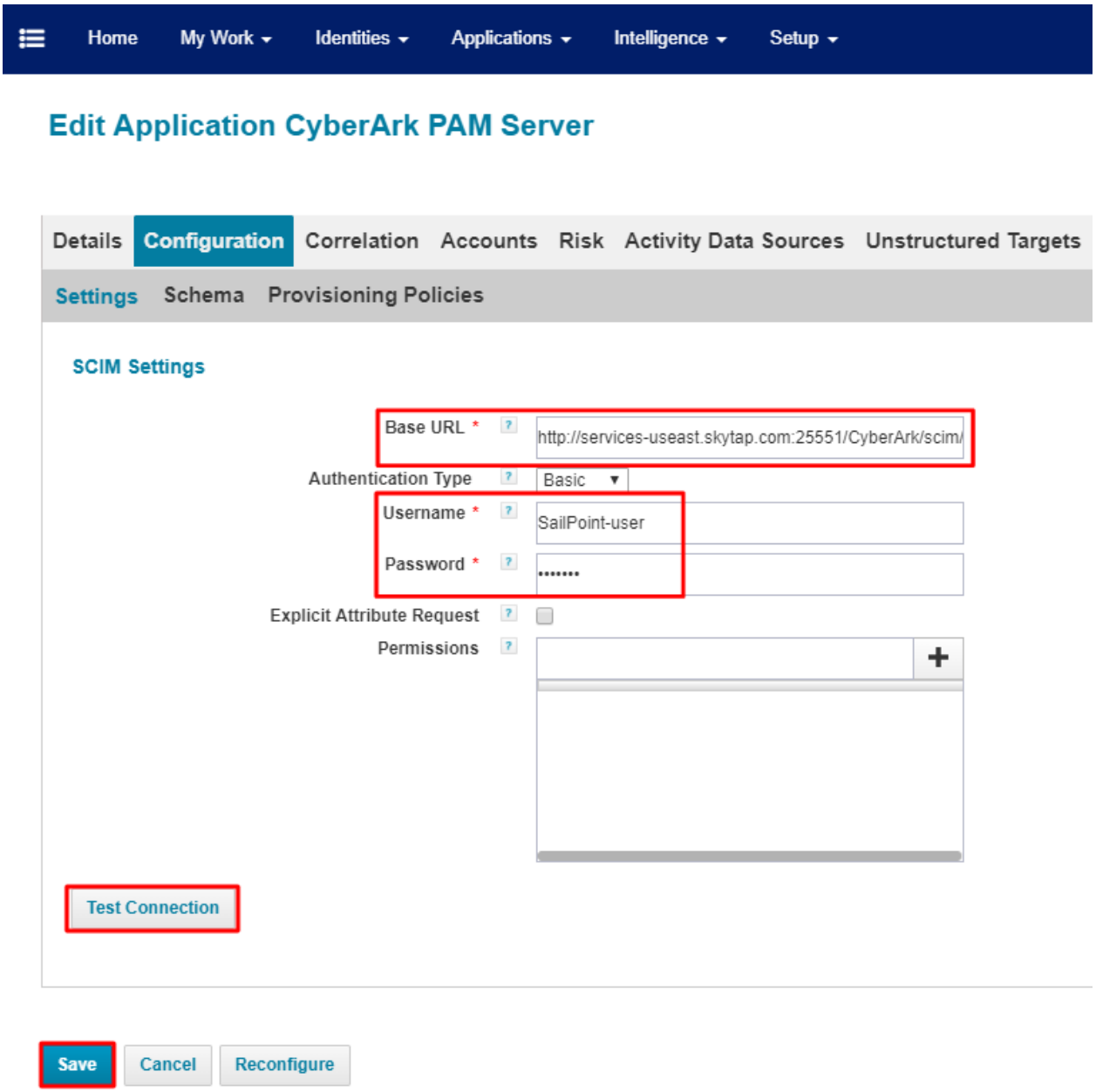


3. Locate and Select **CyberArk PAM Server**
4. Click on the **Configuration** Tab and in the SCIM Settings page enter the following information:

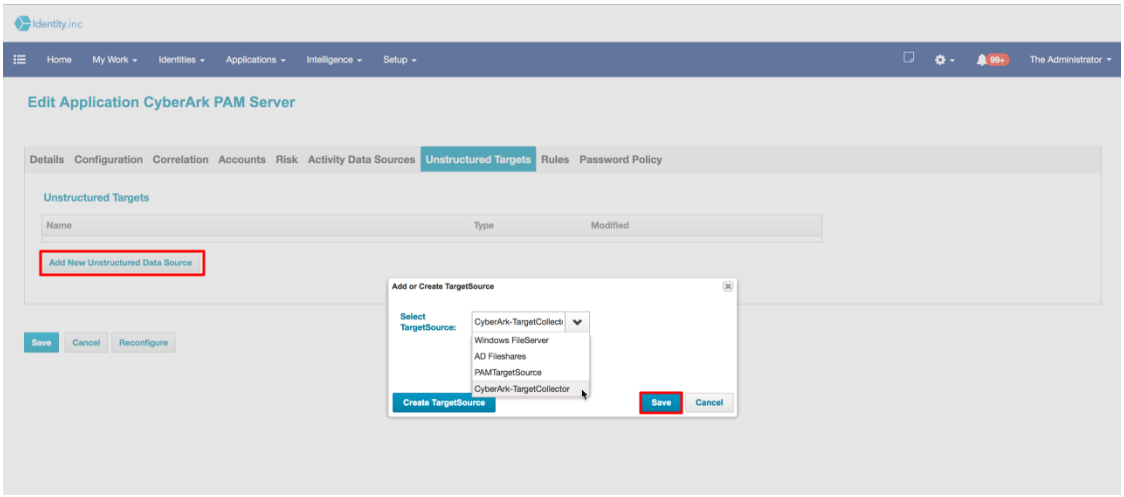
   Base URL: http://<SCIM Server address>:8080/CyberArk/scim/v2
   Username: SailPoint-user

Password: SailPoint-user password (This is the password you provided to the SCIM installer).

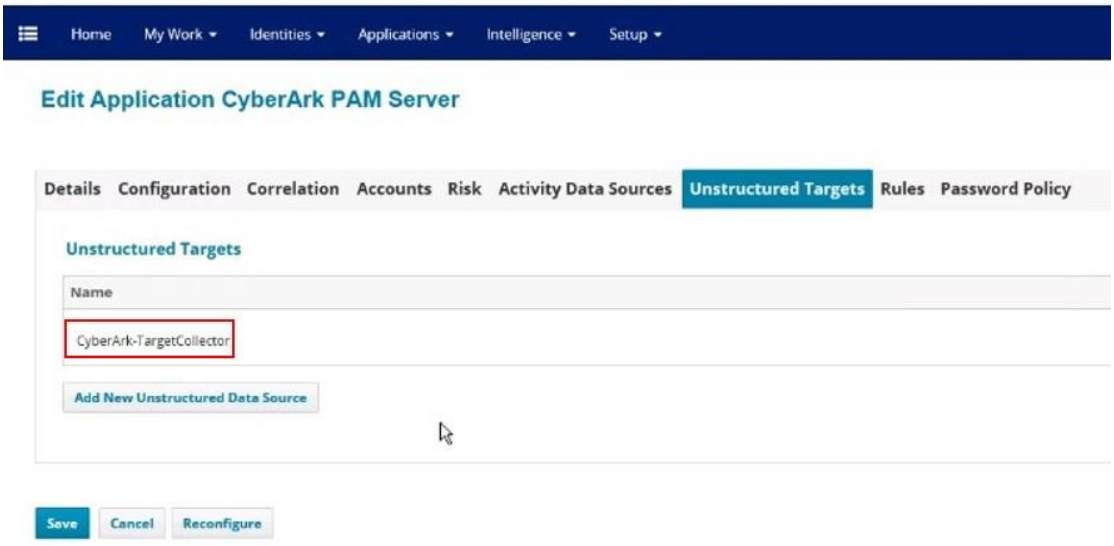
5. Click on **Test Connection** to verify the connection to the SCIM Server
6. If you get a successful Test, Select **Save**.



7. Next Click on the **Unstructured Targets** Tab
8. Select **Add new Unstructured Data Source**
9. In the **Add or Create Target Source** window, Select **CyberArk-TargetCollector** from the pulldown then Select **Save:**



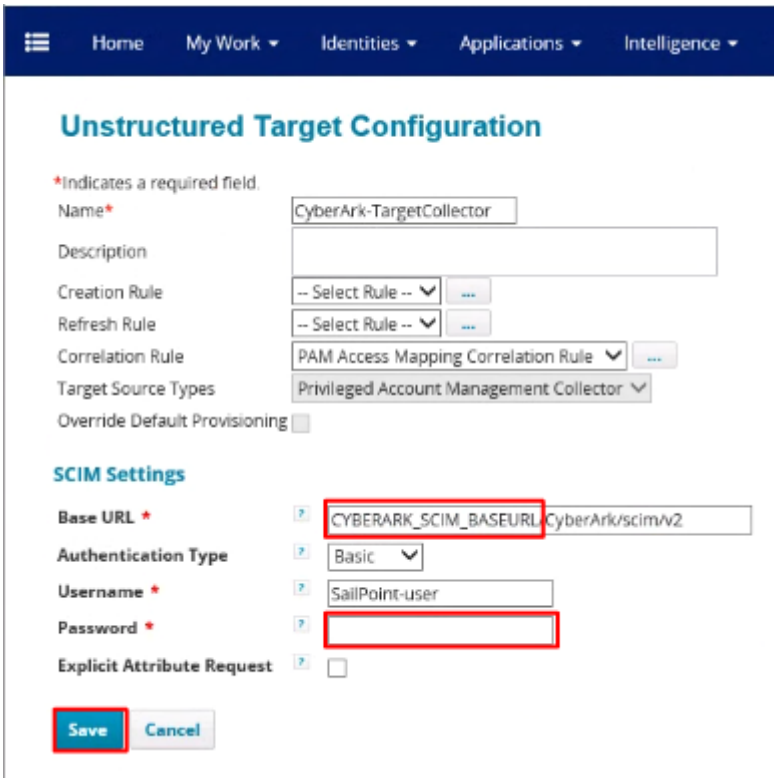
10. In the **Edit Application CyberArk PAM Server** screen Click the **CyberArk-Target Collector**:



11. In the Unstructured Target Configuration window, enter the following parameters:

    Base URL: http://<SCIM Server address>:8080/CyberArk/scim/v2
    Username: SailPoint-user
    Password: SailPoint-user password

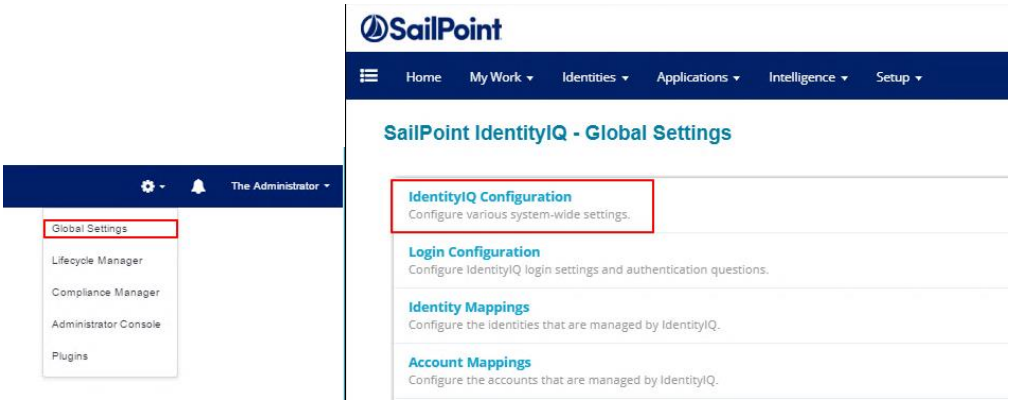
12. Click **Save:**



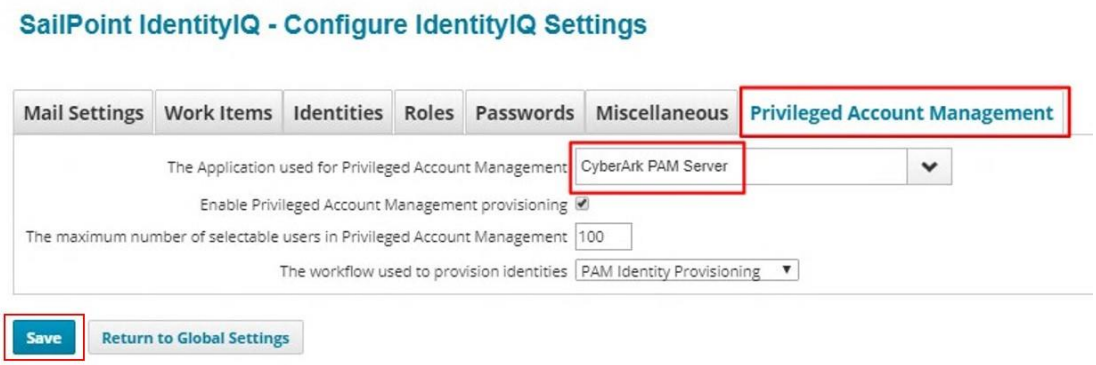
13. In the **Edit Application CyberArk PAM Server** screen Click **Save**

Select the CyberArk PAM Server in IIQ Configuration

1. Click the Settings icon in the top banner and Select **Global** Settings
2. In the **SailPoint IdentityIQ Global Settings** window Select **IdentityIQ Configuration:**

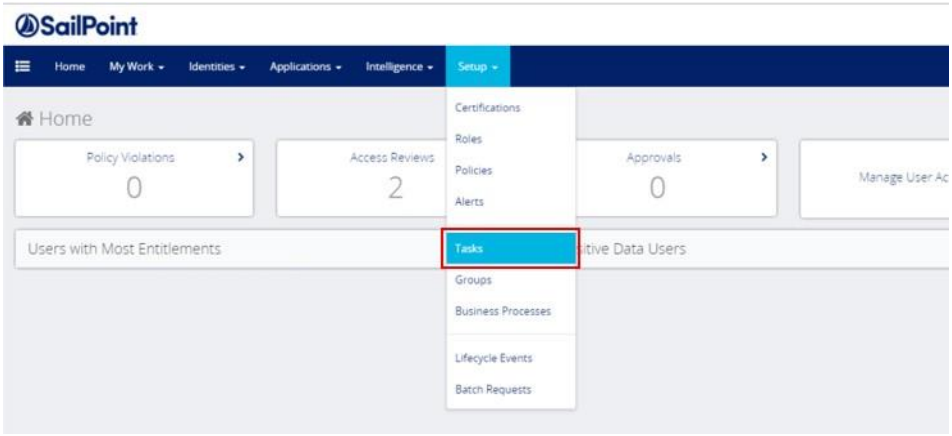


3. This brings up the **SailPoint IdentityIQ – Configure IdentityIQ Settings** window Select the **Privileged Account Management** Tab
4. In the **Application used for Privileged Account Management** parameter Click the dropdown and Select the **CyberArk PAM Server** module
5. Click **Save**:

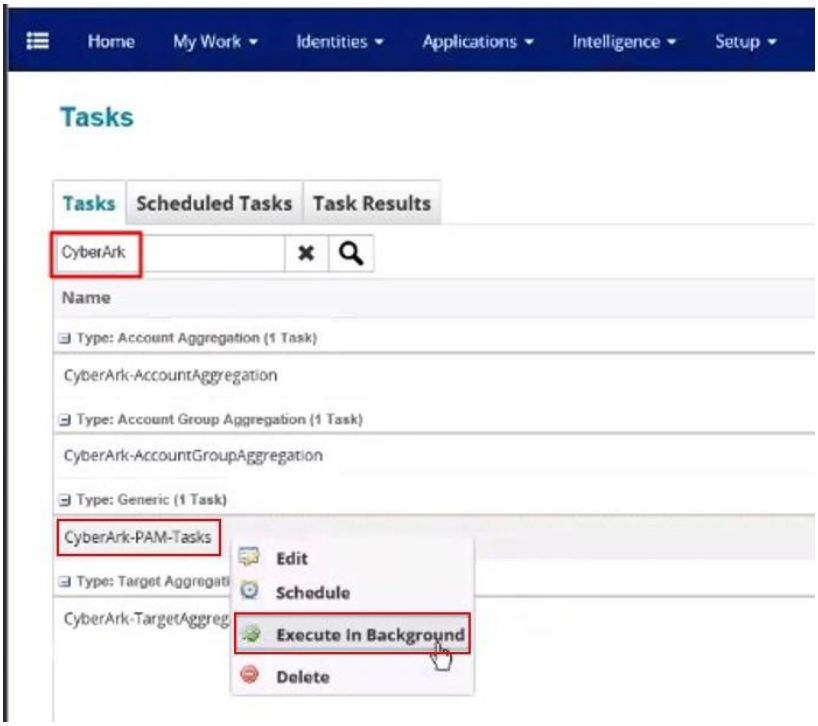


First Run

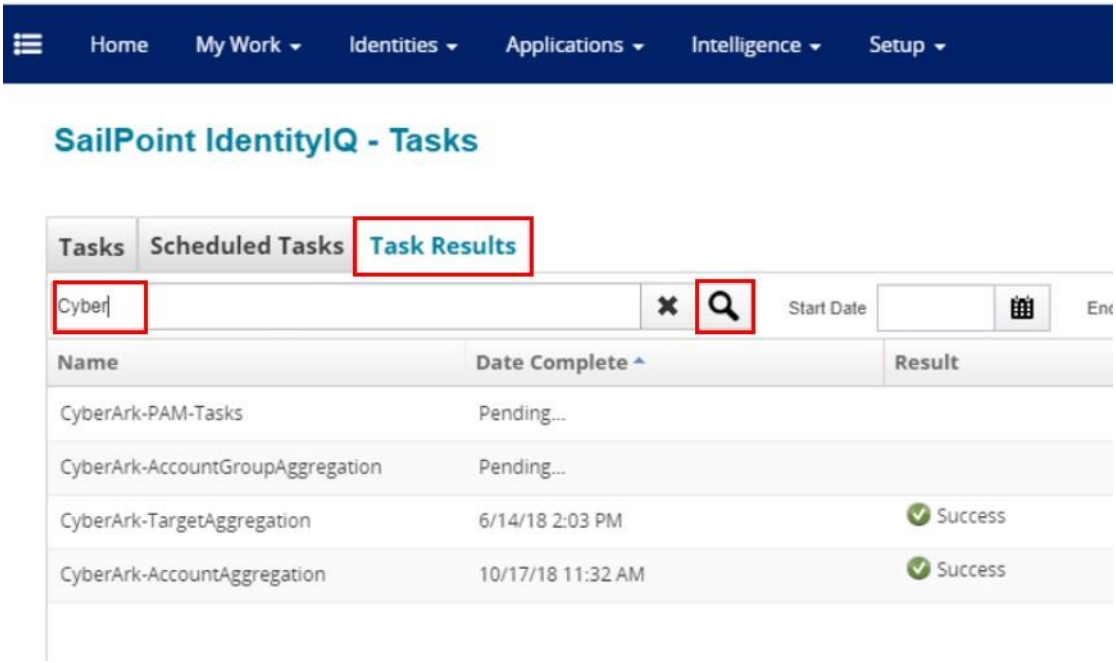
1. Authenticate to SailPoint IIQ and Click the **Setup** Tab; then Select **Tasks**



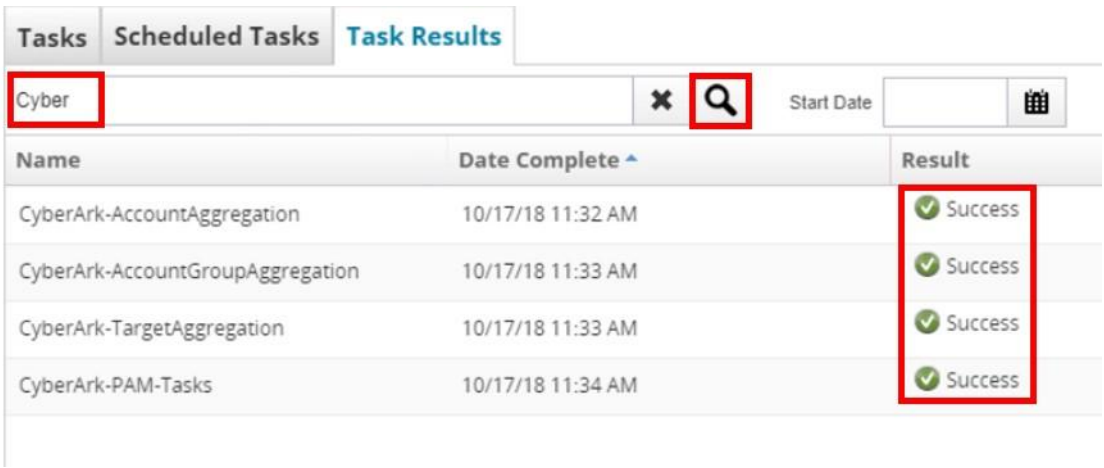
2. In the Tasks window, Search for **CyberArk**
3. Right-Click **CyberArk-PAM-Tasks** and Select **Execute in Background**:



Click on the **Task Results** Tab and enter a search for **CyberArk**



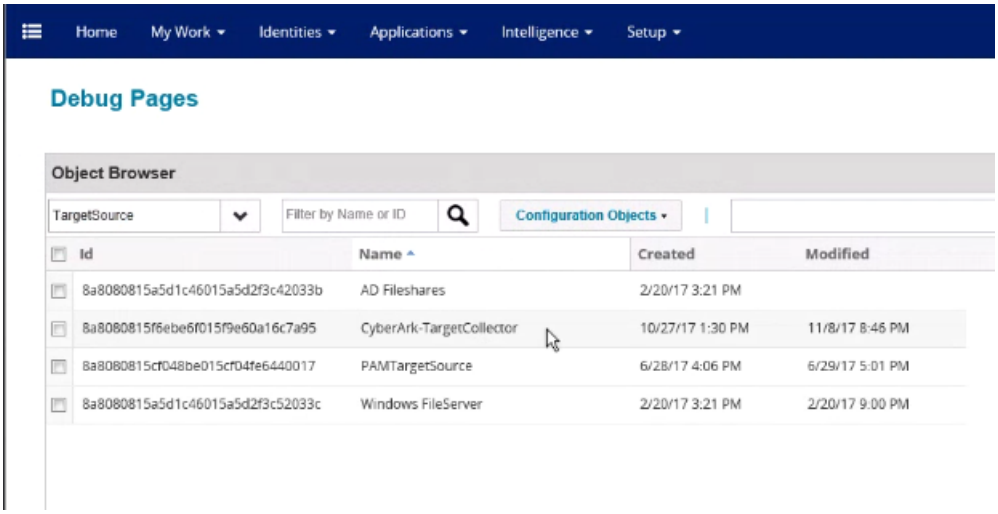
This is what a successful execution looks like:



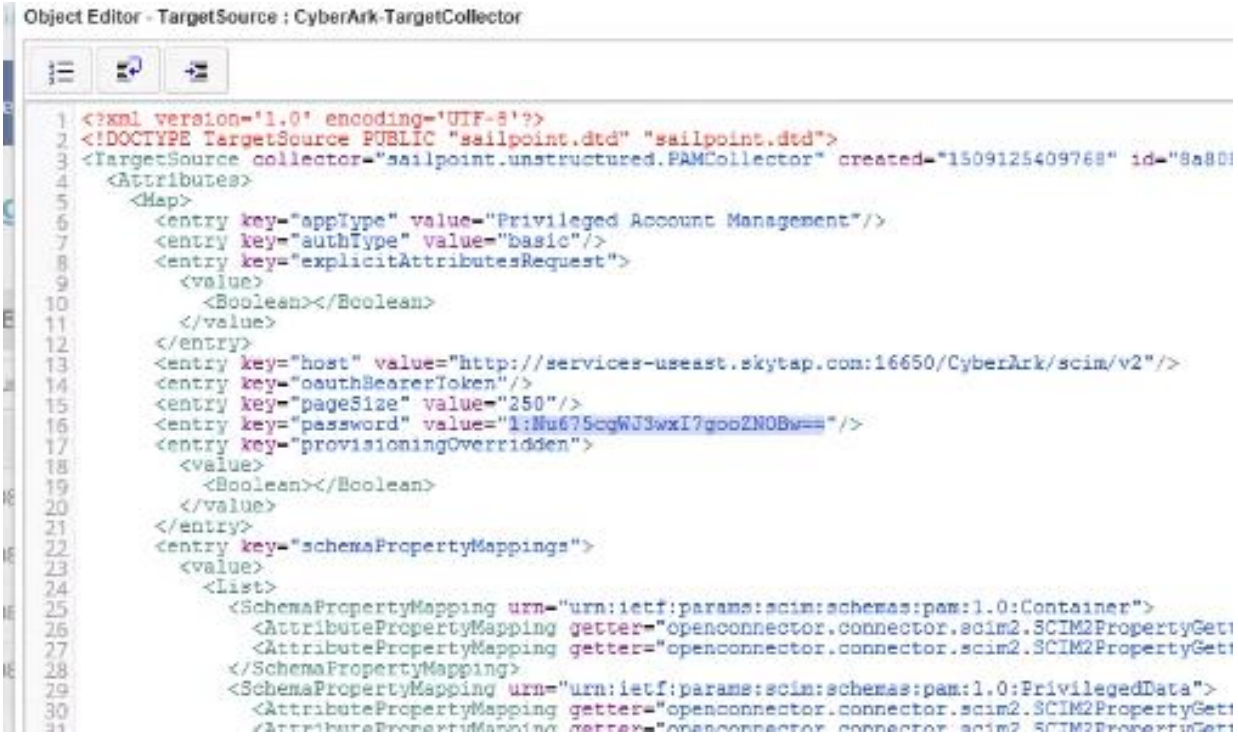
If the **CyberArk-TargetAggregation** task fails, you'll need to replace the password for the **SailPoint-user** with its unencrypted value. To do this, go into **IdentityIQ debug mode** by navigating to the following address in your browser:

https://<IdentityIQ BASEURL>identityiq/debug

Once there, search for **CyberArk-TargetCollector** in the Object Browser:



This brings you to the **Object Editor**, look for the **key=password** entry and replace the encrypted password value for the **SailPoint-user** with its unencrypted value:



Then Save, exit debug mode, and re-run the **CyberArk-PAM-Tasks**.

SailPoint IdentityNow Technical Overview

Use Cases

Use Case 1: CyberArk Account and Group Governance

IdentityNow leverages a SCIM based bi-directional connector to aggregate and provision the user accounts and group assignments within CyberArk.

From an aggregation standpoint, the connector aggregates user accounts and groups from CyberArk. The user accounts and any associated (assigned) groups, are then correlated to identities, and applied to governance processes, such as certification campaigns, or SoD policies.

From a provisioning standpoint, the connector can be configured to:

1. Create user accounts in CyberArk
2. Modify user accounts through group assignments or removals, or through attribute sync processes designed to keep account attributes relevant
3. Enable or disable user accounts through configuration of lifecycle states

Use Case 2: CyberArk Container Permissions Governance

IdentityNow leverages a SCIM based bi-directional connector to aggregate, add, or remove any container (i.e. safe) permissions which might be associated to CyberArk user accounts under governance.

From an aggregation standpoint, the connector aggregates user accounts and any container permissions from CyberArk via SCIM PAM extensions. The user accounts and any associated (assigned) container permissions, are then correlated to identities, and applied to governance processes, such as certification campaigns, or SoD policies.

Use Cases not in Scope

There are a few additional use cases that well understood and currently not addressed by this Integration. These are well understood and will be prioritized higher driven by shift in market pressure.

CyberArk Container (i.e. Safe) Management

IdentityNow can manage the permissions granted to a particular container and user account as part of use case 2. However, IdentityNow does not manage the creation and lifecycle for CyberArk Containers (i.e. Safes) themselves. In order to manage these objects, we recommend using CyberArk's administrative interface as it is intended.

CyberArk Safes for IdentityNow Sources

IdentityNow leverages source service account credentials which are encrypted and stored in IdentityNow. When these are needed by a connector for aggregations or provisioning, the encrypted password is communicated to the virtual appliance where

they are decrypted and sent to the connector for connecting to target sources. Currently, it is not possible to leverage credentials in CyberArk Safes for these source communications.

Stay tuned to the IdentityNow product roadmap for potential developments in these areas. If you have interest in one of these use cases, please contact your IdentityNow Customer Success Manager (CSM).

Technical Details

Use Case 1: CyberArk Users and Groups Governance

The core implementation of this use case is powered by a SCIM 2.0 source configuration.

For aggregation, the following steps must be followed:

1. As an **administrator**, login to IdentityNow.
2. Go to **Admin** -> **Connections** -> **Sources**
3. Click **+New** and enter the following details

Name	Value
Source Type	SCIM 2.0
Source Name	CyberArk Users and Groups (or whatever you prefer)
Description	CyberArk Users and Groups (or whatever you prefer)
Connection Type	Direct Connection

4. Click **Continue**.
5. On the **Config** tab, configure:
    a. **Source Owner**
    b. **Virtual Appliance Cluster**
    c. Used For enable Accounts
    d. Connection Settings - either OAuth 2.0 or Basic Authentication
    e. Server Host with the SCIM 2.0 URL - e.g. http://{host}/CyberArk/scim/v2
6. Click **Save**; then **Test Connection**.

    NOTE: If you cannot connect, check the configuration settings in IdentityNow and CyberArk, as well as any networking (e.g. firewall, DNS, etc.) settings you might have. These are common areas for failure.

7. On the **Import Data** tab, go to **Account Schema**
8. Verify your account schema attributes. Leverage the Discover button to discover any new attributes from CyberArk. Do not change the Account ID, Name, or Entitlement attributes.

9. On the **Import Data** tab, go to **Correlation**.
10. Define your account correlation mappings between the CyberArk user accounts and the IdentityNow identity model. Your data may vary, but we recommend:
    a. Work Email Identity Attribute equals emails.work.primary.value Account Attribute
    b. User Name Identity Attribute equals userName Account Attribute
11. On the **Import Data** tab, go to **Entitlement Aggregation**.
12. Click **Start** to aggregate groups from CyberArk.
13. On the **Import Data** tab, go to **Account Aggregation**.
14. Click **Start** to aggregate users and group assignments from CyberArk.

For provisioning, the additional steps must be followed:

1. On the **Config** tab, configure:
    a. **Used For** also enable **Provisioning** and **Password Management**.
2. On the **Accounts** tab, go to **Create Profile**.
3. Define the attribute mappings specific to your CyberArk accounts.
4. For role-based provisioning, add any CyberArk groups to roles in the system.
5. For account enables or disables, opt this CyberArk source into lifecycle state configurations.
6. For attribute synchronization, configure **Account Sync** under the **Accounts** tab.

Use case 2: CyberArk Container Permissions Governance

From an aggregation standpoint, the connector aggregates user accounts and any container permissions from CyberArk via SCIM PAM extensions. The user accounts and any associated (assigned) container permissions, are then correlated to identities, and applied to governance processes, such as certification campaigns, or SoD policies. IdentityNow represents the permissions on a container in the form:

{container} - {permission}

For example:

ActiveDirectory - Retrieve

From a provisioning standpoint, the connector is primarily designed to handle container permission assignments or removals. If user account creation, attribute modification, or enabling or disabling of the user accounts is needed, this can be accomplished through use case 1.

Implementation

The core implementation of this use case is accomplished by deploying a SCIM PAM connector developed by SailPoint Professional Services.

NOTE: In order to install the connector, follow the instructions which come with the SCIM PAM connector from SailPoint Professional Services. At high level this involves creation and installation of the SCIM PAM connector via IdentityNow REST APIs. This is considered a pre-requisite and must be done before the following configurations can be carried out.

For aggregation, the following steps must be followed:

1. As an **administrator**, login to **IdentityNow**.
2. Go to **Admin** -> **Connections** -> **Sources**.
3. Click **+New** and enter the following details

Name	Value
Source Type	SCIM PAM
Source Name	CyberArk Safes (or whatever you prefer)
Description	CyberArk Safes (or whatever you prefer)
Connection Type	Direct Connection

4. Click **Continue**
5. On the **Config** tab, configure:
    a. **Source Owner**
    b. **Virtual Appliance Cluster**
    c. **Used For** enable **Accounts**
    d. **Authentication** - either **OAuth 2.0** or **Basic Authentication**
    e. **Settings** with the **Base URL**: e.g. http://{host}/CyberArk/
6. Click **Save**, then **Test Connection**.

    NOTE: If you cannot connect, check the configuration settings in IdentityNow and CyberArk, as well as any networking (e.g. firewall, DNS, etc.) settings you might have. These are common areas for failure.

7. On the **Import Data** tab, go to **Account Schema**.
8. Validate the account schema attributes. Do not change the Account ID, Name, or permissions attributes.
9. On the **Import Data** tab, go to **Correlation**.
10. Define your account correlation mappings between the CyberArk user accounts and the IdentityNow identity model. Your data may vary, but we recommend:
    a. Work Email Identity Attribute equals emails.work.primary.value Account Attribute
    b. User Name Identity Attribute equals userName Account Attribute
11. On the **Import Data** tab, go to **Account Aggregation**

12. Click **Start** to aggregate container permission assignments from CyberArk.

Other Considerations

If you are performing aggregations and they fail due to timeout, you can increase the SailPoint API Timeout to allow for extra time if certain cache items need to be built as the calls are made.

Below is an example on how to increase the SailPoint API Timeout to 10 minutes through SailPoint's "**application.xml**" file with the following entry:

```
<entry key="customTimeout" value="10"/>
```

Directory Users and Groups - CyberArk can be configured to leverage directory (e.g. Active Directory) users and their group assignments to automatically tie directory groups to CyberArk group assignments. Since IdentityNow can manage both the directory and CyberArk directly, architects of the IdentityNow and CyberArk solutions recommend adopting a unified approach and deliberately planning the means and method by which CyberArk users and groups are to be provisioned.