Creating an Adapter to Transfer Inventory from a SaaS Product

FlexNet Manager Suite 2018 R1

March, 2018

Contents

Introduction

Version: *These instructions apply to FlexNet Manager Suite 2018 R1 and later.*

This document describes how to use FlexNet Manager Suite to create a PowerShell-based adapter to transfer inventory data from a SaaS product into FlexNet Manager Suite. Okta.com is used as the example SaaS product in this document.

Note: *FlexNet Manager Suite includes a built-in adapter for Salesforce. This document is intended for help creating additional adaptors to transfer inventory data from other SaaS products.*

Before you begin, you will need to:

- Understand the API available from the SaaS product and the data that can be retrieved
- Populate the FlexNet Manager Suite data model

Understand the API available from the SaaS product and the data that can be retrieved

As part of these instructions, you will need to refer to the API reference of your SaaS product in order to understand:

- Details about how to call the APIs
- Details about the authentication method
- Details regarding the kinds of data that can be retrieved. At a minimum, user accounts and access evidence (most likely in the form of applications) are required. Knowledge of how to retrieve the license data will be helpful.

Populate the FlexNet Manager Suite data model

The following FlexNet Manager Suite objects are recommended for SaaS license modeling.

Table 1: Data model requirements and recommendations for SaaS adapters

FlexNet Manager Suite Object	Description	Requirement
Computer	There will be one computer representing each SaaS instance	Required
ClientAccessEvidence	Access evidence; this evidence can be in the form of SaaS applications	Required
AccessingUser	User accounts in a SaaS product	Required
ClientAccessedAccessEvidence	List of user accounts that accessed each SaaS application	Recommended
ClientAccessedAccessOccurrence	Any occurrences of a user accessing each SaaS application	Recommended
SoftwareTitle	SaaS applications in a SaaS product	Recommended
SoftwareLicense	SaaS license(s)	Recommended
SoftwareTitleLicense	SaaS applications that belong to each SaaS license	Recommended
SoftwareTitleAccessEvidence	Mapping of access evidence to SaaS application	Recommended
SoftwareLicenseAllocation	Users allocated to SaaS licenses	Recommended

If data is not available to fill the recommended tables, you will need to decide how to model the data from the data that is available. For example, the Okta API does not contain licensing information, so we model licenses using the application data such that we have one license for each application.

Creating the adapter

To create the adapter, you will need:

- Reader XML files
- PowerShell script
- Reader config file
- Correct placement of adapter files

Reader XML files

Create a reader XML file to define the mapping to the FlexNet Manager Suite tables. You need to define Reader steps for each of the FlexNet Manager Suite tables provided in the FlexNet Manager Suite data model section.

Reader steps are of type **SourceToObject**. This indicates that the data from the source will be sent to the data object defined. The **Language** property should be set to **PowerShell**. The **Method** property should be set to a PowerShell method name.

Table 2: DataObject to FlexNet Manager Suite table mapping

DataObject	FlexNet Manager Suite Table
Computer	ImportedComputer
ClientAccessEvidence	ImportedClientAccessEvidence
AccessingUser	ImportedAccessingUser
ClientAccessedAccessEvidence	ImportedClientAccessedAccessEvidence
ClientAccessedAccessOccurrence	ImportedClientAccessedAccessOccurrence
SoftwareTitle	ImportedSoftwareTitle
SoftwareLicense	ImportedSoftwareLicense
SoftwareTitleLicense	ImportedSoftwareTitleLicense
SoftwareTitleAccessEvidence	ImportedSoftwareTitleAccessEvidence
SoftwareLicenseAllocation	ImportedSoftwareLicenseAllocation

Refer to the **InventoryObjectModel.xml** (by default, located in the C:\ProgramData\Flexera Software\Compliance\ImportProcedures\ObjectAdapters folder) for more details on all available data objects and their corresponding FlexNet Manager Suite database tables.

The parameters for the methods should be declared using the **InputVariable** XML element. The method's parameter values are set to the values retrieved from the Inventory Beacon. For more information, refer to the Reader config file section of this document.

Table 3: InputVariable tags

InputVariable Attribute	Description
Name	PowerShell method's parameter name
Variable	Parameter defined in the reader config

The mapping of the properties from the source object to the fields of the database tables are indicated by the **OutputSchema** tags.

Table 4: OutputSchema tags

OutputSchema Attribute	Description
Name	Name of the column in the FlexNet Manager Suite table
Property	Name of the property in the PowerShell object returned from the method
Type	SQL type
Length	Length of (n)varchar (Only required for (n)varchar values)

In the FlexNet Manager Suite data model, a SaaS instance is represented by a computer. The first Reader step should be to create that computer. All data from the SaaS instance is associated with this computer.

In our Okta example, the source PowerShell method is **Get-OktaHostName**. The only input parameter is **oktaUrl** and its value is set from the user's input on the Inventory Beacon.

```
<Reader xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xsi:type="SourceToObject"
        Name="Create host to link all records"
        Order="110"
        Retries="1"
        Language="PowerShell"
        Method="Get-OktaHostName"
        DataObject="Computer">

        <InputVariable Name="oktaUrl" Variable="oktaUrl" />

        <OutputSchema Name="ExternalID" Type="int" Property="ExternalID" />
        <OutputSchema Name="ComputerName" Type="nvarchar" Length="256"
        Property="ComputerName" />
        <OutputSchema Name="Domain" Type="nvarchar" Length="100" Property="Domain" />
        <OutputSchema Name="SerialNo" Type="nvarchar" Length="100" Property="ExternalID" />
        <OutputSchema Name="InventoryAgent" Type="nvarchar" Length="256"
        Property="InventoryAgent" />
        <OutputSchema Name="UntrustedSerialNo" Type="bit" Property="UntrustedSerialNo" />
        <OutputSchema Name="IsRemoteACLDevice" Type="bit" Property="IsRemoteACLDevice" />
</Reader>
```

The **Get-OktaHostName** PowerShell method is shown in the following code. It takes one parameter as defined by the **InputVariable** tag in the Reader step above. The method returns an object with certain properties.

```
function Get-OktaHostName($oktaUrl)
{
        [PSCustomObject]@{
                ExternalID=1
                ComputerName=$oktaUrl
                Domain=''
                SerialNo='serial: ' + $oktaUrl
                InventoryAgent='Okta'
                UntrustedSerialNo=$false
                IsRemoteACLDevice=$false
        }
}
```

This next Reader step transfers user data to the **ImportedAccessingUser** table.

The source for the data is the **Get-Users** PowerShell method. The method takes two parameters (**oktaUrl** and **apiToken**) as indicated by the **InputVariable** tags.

For each user, we retrieve the **UserId**, **EmailUserName**, and **EmailDomainName**. That data is placed in the **ExternalAccessingUserID**, **UserName**, **DomainName**, and **SAMAccountName** fields of the

ImportedAccessingUser temp table; where **EmailUserName** is used to populate both the **UserName** and **SAMAccountName** fields.

```
<Reader xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xsi:type="SourceToObject"
        Name="Get Okta users for recording access to apps later"
        Order="130"
        Retries="1"
        Language="PowerShell"
        Method="Get-Users"
        DataObject="AccessingUser">

        <InputVariable Name="oktaUrl" Variable="oktaUrl" />
        <InputVariable Name="apiToken" Variable="apiToken" />

        <OutputSchema Name="ExternalAccessingUserID" Type="nvarchar" Length="256"
        Property="UserId" />
        <OutputSchema Name="UserName" Type="nvarchar" Length="256"  Property="EmailUserName"
        />
        <OutputSchema Name="DomainName" Type="nvarchar" Length="100"
        Property="EmailDomainName" />
        <OutputSchema Name="SAMAccountName" Type="nvarchar" Length="64"
        Property="EmailUserName" />
</Reader>
```

The **Get-Users** method is shown in the following example. The parameters of the method correspond to the **InputVariable** tags of the Reader step. A result object is constructed with the properties that are needed as indicated by the **OutputSchema** tags.

```
function Get-Users($oktaUrl, $apiToken)
{
        $action = '/api/v1/users'
        $result = Invoke-OktaRestRequest $oktaUrl $apiToken $action

        $result | ForEach-Object { $_ |
                Select-Object -Property @{N='UserId'; E={$_.id}},
                        @{N='Email'; E={$_.profile.email}},
                        @{N='EmailUserName'; E={$_.profile.email.split('@')[0]}},
                        @{N='EmailDomainName'; E={$_.profile.email.split('@')[1]}},
                        @{N='MapUsingEmailAddress'; E={$true}}
            }
}
```

The accompanying example **License.xml** file contains the Reader steps for mapping the Okta data to the FlexNet Manager Suite tables.

PowerShell script

You will need to create a PowerShell script to use the **Invoke-WebRequest** PowerShell method which will invoke the SaaS API.

Note: *Using PowerShell version 3.0 is preferred.*

You will need to refer to your SaaS product's documentation regarding how to invoke their API. The API calls you make may not work without an authentication token.

For our example, Okta requires that you log on to the site to generate an API token. This token will be needed for each call to an API. The Okta API requires the API token to be set in the header of the web request. The **Authorization** property should be set to **'SSWS'** which is appended with the API token.

The sample shown in this section provides a function that invokes a given Okta API via **Invoke-WebRequest**. It takes three parameters:

- The URL to an Okta instance
- The API token
- The API to call

The API is continuously called until all the data has been fetched at which point the data is stored into a result object.

```
function Invoke-OktaRestRequest($oktaUrl, $apiToken, $action)
{
	$headers = @{
		Accept="application/json"
		"Content-Type"="application/json"
		Authorization='SSWS ' + $apiToken
	}

	$url = $oktaUrl + $action

	$result = Invoke-WebRequest -Uri $url -Method GET -Headers $headers

	$result | ConvertFrom-Json # dump current results to pipe

	$link = $result.Headers.Link

	while ($link -Match '; rel="next"') {
		# Pagination required
		$url = $link -Split 'rel="self",<', 0, 'simplematch'
		$url = $url[1] -Split '>; rel="next"'

		$nextPage = $url[0]

		$result = Invoke-WebRequest -Uri $nextPage -Method GET -Headers $headers

		$result | ConvertFrom-Json # dump current results to pipe

		$link = $result.Headers.Link
	}
}
```

The Okta API for retrieving users is **'/api/v1/users'**. The **Get-Users** method uses the **Invoke-OktaRestRequest** helper method.

```
function Get-Users($oktaUrl, $apiToken)
{
	$action = '/api/v1/users'
	$result = Invoke-OktaRestRequest $oktaUrl $apiToken $action

	$result | ForEach-Object { $_ |
		Select-Object -Property @{N='UserId'; E={$_.id}},
			@{N='Email'; E={$_.profile.email}},
			@{N='EmailUserName'; E={$_.profile.email.split('@')[0]}},
			@{N='EmailDomainName'; E={$_.profile.email.split('@')[1]}},
			@{N='MapUsingEmailAddress'; E={$true}}
		}
}
```

The following sample provides another example of a PowerShell method. Since Okta does not have an API for retrieving license information, this example creates a license for each application employing the **Get-LicensesToGenerate** method.

```
function Get-LicensesToGenerate($oktaUrl, $apiToken)
{
	$action = '/api/v1/apps'
	$result = Invoke-OktaRestRequest $oktaUrl $apiToken $action

	$result | ForEach-Object { $_ |
		Select-Object -Property @{N='LicenseId'; E={$_.id}}, # use AppId
				@{N='LicenseName'; E={$_.label}},
				@{N='SoftwareLicenseTypeID'; E={42}}, # SaaS User license type
				@{N='EntitlementCount'; E={0}},
				@{N='IsSubscription'; E={$false}}
		}
}
```

***Note**: Licenses created should be of type **SaaS User**, **SoftwareLicenseTypeID = 42**.*

The accompanying example **Logic.ps1** file contains all the PowerShell methods for retrieving the data from Okta.

Reader config file

The reader config file should be named **readerV3.config**. The file describes: the input fields to display in the Inventory Beacon connection dialog, the PowerShell scripts to load, the method to retrieve version information, and the ordered list of Reader XML files to use.

The **Configuration** tag should be set as shown in the example in this section. Note the following:

- The **Parameters** tag describes the fields to display in the Inventory Beacon for the user to fill-in. These fields are the parameters needed for the Okta PowerShell methods.
- The **InitScripts** indicates the PowerShell scripts to load.
- The **VersionCheck** tag indicates the PowerShell method to call to get the version of the adapter. Since the **Get-Version** method for this Okta example needs parameters, the parameters are described. Each Reader step can have a **VersionTo** and/or a **VersionFrom** property defined to indicate upon which versions of the adapter the step should execute.
- The **Readers** tag indicates the execution order of the Reader xml files.

```
<?xml version="1.0" encoding="utf-8"?>
<Configuration Version="3.0" Language="PowerShell">
	<Parameters>
		<Parameter Name="oktaUrl" DisplayText="Okta URL" Type="text"
IsMandatory="true" />
		<Parameter Name="apiToken" DisplayText="API Token" Type="password"
IsMandatory="true" />
	</Parameters>

	<InitScripts>
		<InitScript Name="Logic.ps1" />
	</InitScripts>

	<VersionCheck Language="PowerShell" Method="Get-Version">
		<Parameter Name="oktaUrl" Variable="oktaUrl" />
		<Parameter Name="apiToken" Variable="apiToken" />
	</VersionCheck>

	<Readers>
		<Reader FileName="License.xml" ProcedureName="LicenseCreateUpdate" />
	</Readers>
</Configuration>
```

Using the adapter with the Inventory Beacon

The adapter files need to be placed in the correct location so that the Inventory Beacon will pick it up.

To place the adapter files in the correct location:

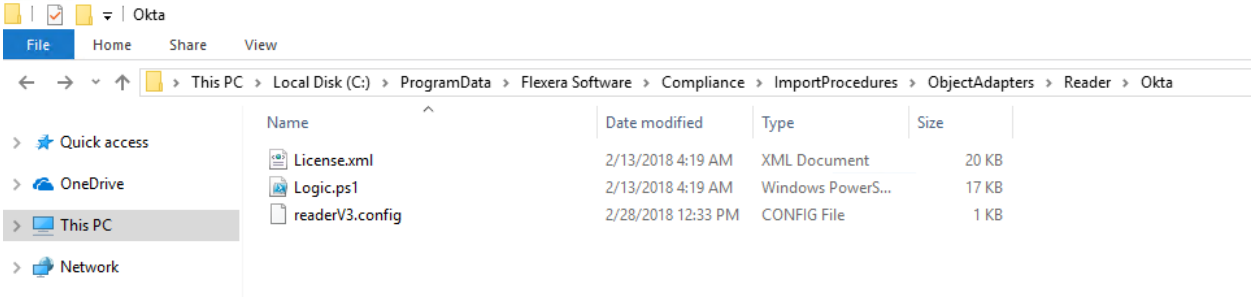
1. Locate the **ObjectAdapters** folder, typically located at:

   `C:\ProgramData\Flexera Software\Compliance\ImportProcedures\ObjectAdapters`

2. Make sure there is a **Reader** folder in the ObjectAdapters folder; if not, create it.

3. Create a new folder in the Reader folder and place the PowerShell scripts, Reader xml files, and **readerV3.config** file in that folder.

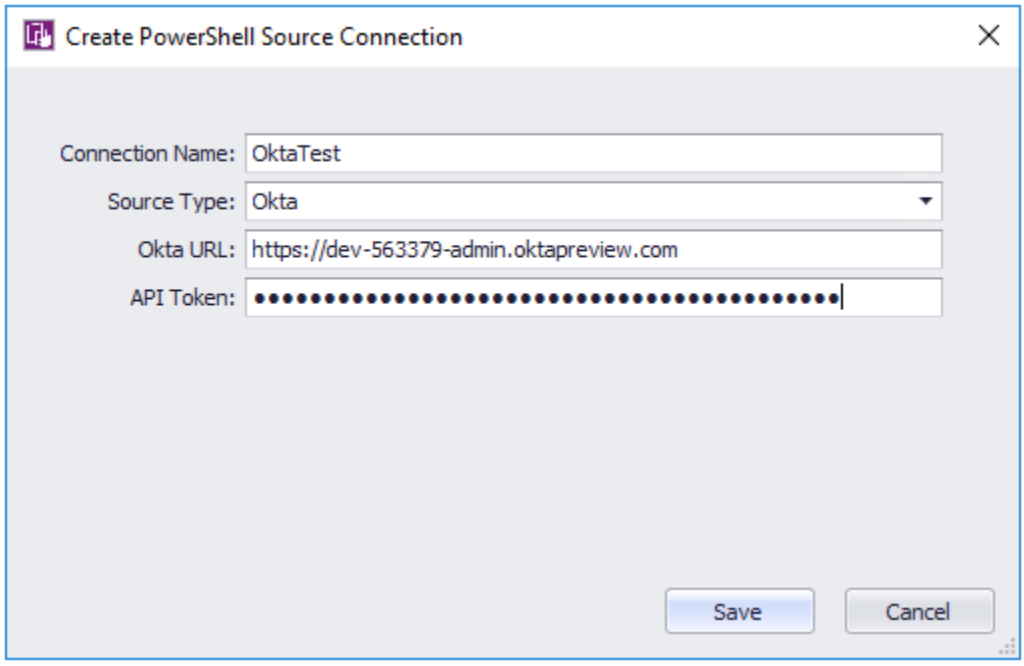
Placement of files as shown:



To create a connection with the adapter:

1. Launch the Inventory Beacon.
2. Click the **New…** button to create a new PowerShell connection.
3. Create a new folder in the Reader folder and place the PowerShell scripts, Reader xml files, and **readerV3.config** file in that folder.

The following shows the Beacon, when creating a new PowerShell connection with **Source Type** set to **Okta**:



With a successful connection configured, FlexNet Manager Suite is now ready to import inventory from Okta.

About Flexera

Flexera is reimagining the way software is bought, sold, managed and secured. We view the software industry as a supply chain, and make the business of buying and selling software more transparent, secure, and effective. Our Monetization and Security solutions help software sellers transform their business models, grow recurring revenues and minimize open source risk. Our Vulnerability and Software Asset Management (SAM) solutions strip waste and unpredictability out of buying applications, helping companies purchase only the software and cloud services they need, manage what they have, and reduce license compliance and security risk. In business for 30+ years, our 1000+ employees are passionate about helping our 80,000+ customers generate millions in ROI every year. Visit us at: www.flexera.com

Flexera

300 Park Blvd., Suite 500

Itasca, IL 60143

USA

Itasca (Global Headquarters):

+1 800-374-4353

United Kingdom (Europe, Middle East Headquarters)

+44 370-871-1111

+44 870-873-6300

Japan (Asia, Pacific Headquarters)

+81 3-4360-8291

Australia

+61 3 9895 2000

www.flexera.com