Dynatrace-Salesforce Integration - Salesforce Documentation

Table of contents

Overview

The extension reports Real User Sessions and Real User Actions. The data is captured from the Salesforce Event Streaming API.

Properties captured per event

The properties below are captured for the corresponding events, they appear in the user action waterfall. Options marked with an * are only captured if `Capture user details` is enabled.

Note that you need to create action or session properties to query these in Dynatrace using UQL.

Login	Logout	API	Lightning URI	ListView	Report	URI
• AdditionalInfo • ApiType • ApiVersion • Application • AuthMethodReference • Browser • CipherSuite • ClientVersion • Country • CountryIso • EvaluationTime • EventDate • EventIdentifier • LoginKey • LoginType • Platform • SessionKey • SessionLevel • SourceIp* • Status • UserId • Username* • UserType	• LoginKey • SourceIp • EventDate • EventIdentifier • SessionKey • UserId • Username • ReplayId • SessionLevel	• AdditionalInfo • ApiType • ApiVersion • Application • Client • ElapsedTime • EvaluationTime • EventDate • EventIdentifier • LoginHistoryId • Operation • Platform • PolicyId • PolicyOutcome • QueriedEntities • Query • Records • RelatedEventIdentifier • RowsProcessed • RowsReturned • SessionKey • SessionLevel • SourceIp* • UserAgent • Username* • UserId	• AppName • ConnectionType • DeviceId • DeviceModel • DevicePlatform • DeviceSessionId • Duration • EffectivePageTime • EventDate • EventIdentifier • LoginKey • Operation • OsName • OsVersion • PageStartTime • PageUrl • PreviousPageAppName • PreviousPageEntityId • PreviousPageUrl • QueriedEntities • RecordId • RelatedEventIdentifier • ReplayId • SdkAppType • SdkAppVersion • SdkVersion • SessionKey • SessionLevel • SourceIp* • UserId • Username* • UserType	• AppName • ColumnHeaders • DeveloperName • EvaluationTime • EventDate • EventIdentifier • EventSource • ExecutionIdentifier • FilterCriteria • ListViewId • LoginHistoryId • LoginKey • Name • NumberOfColumns • OrderBy • OwnerId • PolicyId • PolicyOutcome • QueriedEntities • Records • RelatedEventIdentifier • RowsProcessed • Scope • Sequence • SessionKey • SessionLevel • SourceIp* • UserId • Username*	• ColumnHeaders • DashboardId • DashboardName • Description • DisplayedFieldEntities • EventDate • EventIdentifier • EventIdentifier • EventSource • ExecutionIdentifier • ExportFileFormat • GroupedColumnHeaders • LoginHistoryId • LoginKey • NumberOfColumns • Operation • OwnerId • PolicyId • PolicyOutcome • QueriedEntities • Records • RelatedEventIdentifier • ReplayId • ReportId • RowsProcessed • Scheduled • Scope • SessionKey • SessionLevel • SourceIp* • UserId • Username*	• EventDate • EventIdentifier • LoginKey • Message • Name • Operation • OperationStatus • QueriedEntities • RecordId • RelatedEventIdentifier • SessionKey • SessionLevel • SourceIp* • UserId • Username* • UserType

Requirements

Tools required

- openssl
- keytool (comes with a Java installation)
- Salesforce account that can create Connected Apps

Salesforce Requirements

These are the requirements to configure the monitoring from Salesforce point of view

- Profile that has `View Real-Time Event Monitoring Data` user permission.
- Event Streaming enabled in SF. Instructions on how to enable Event Streaming can be found here
- Check that under `Setup > Events Manager` the events have Streaming Data enabled.

An example of an account configured with Event Streaming (`Setup > Events > Event Manager`)

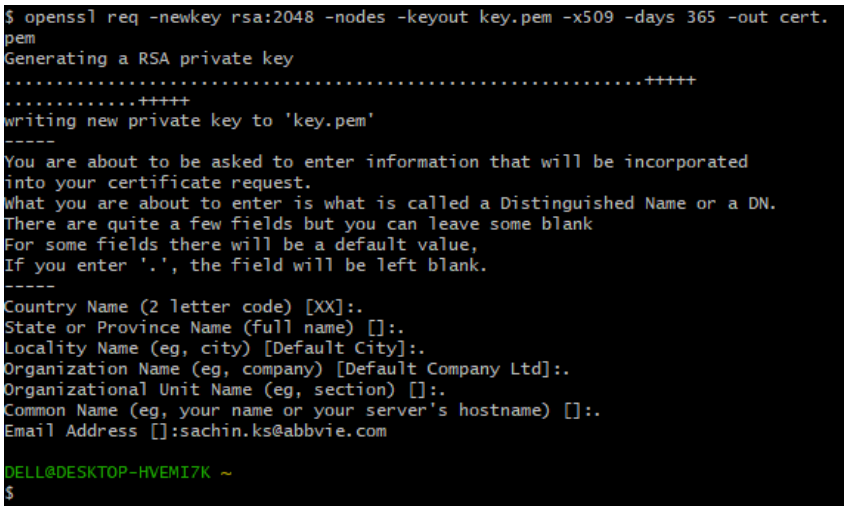
blocked URL

Creating Certificates

The Connected App will need a certificate attached to it. You can use an existing one or create a new certificate yourself, these steps show how to create a certificate using `openssl` and `keytool`

- Step 1 - Create a certificate and it's private key, you can accept all default options

```
openssl req -newkey rsa:2048 -nodes -keyout key.pem -x509 -days 365 -out cert.
pem
```



This will create two files called `cert.pem` and `key.pem`

- Step 2 - Merge both files in one, you can use a text editor, cat, etc.

```
cat cert.pem key.pem >> full_cert.pem
```

After this step you should have a file called `full_cert.pem`

- Step 3 - Add this certificate to a Java Keystore (jks) file, this will later be used in Dynatrace

  Convert the `full_cert.pem` file to pkcs12, you **must** set a password when it asks for one

```
openssl pkcs12 -export -out full_cert.pkcs12 -in full_cert.pem
```

```
DELL@DESKTOP-HVEMI7K ~
$ openssl pkcs12 -export -out full_cert.pkcs12 -in full_cert.pem
Enter Export Password:
Verifying - Enter Export Password:

DELL@DESKTOP-HVEMI7K ~
$
```

Add the file to a new Java Keystore. You must set a password for the keystore (destination password). The source password is the one you created on the previous command

Open cmd from the location where keytool file exists(java folder) in the local system and run the following command. Create destination password and also Key in the source password created in Step 3.


```
keytool -importkeystore -srckeystore "C:\cygwin64\home\DELL\full_cert.pkcs12" -srcstoretype pkcs12 -destkeystore "C:\cygwin64\home\DELL\full_cert.jks" -deststoretype JKS
```

```
C:\Program Files\Java\jdk-16.0.1\bin>keytool -importkeystore -srckeystore "C:\cygwin64\home\DELL\full_cert.pkcs12" -srcstoretype pkcs12 -destkeystore "C:\cygwin64\home\DELL\full_c
Importing keystore C:\cygwin64\home\DELL\full_cert.pkcs12 to C:\cygwin64\home\DELL\full_cert.jks...
Enter destination keystore password:
Re-enter new password:
They don't match. Try again
Enter destination keystore password:
Re-enter new password:
Enter source keystore password:
Entry for alias 1 successfully imported.
Import command completed:  1 entries successfully imported, 0 entries failed or cancelled
```

After all is done you should have five files, and we will only use two of them:

- key.pem
- cert.pem < **Will be used when creating the connected app in salesforce**
- full_cert.pem
- full_cert.pkcs12
- full_cert.jks < **Will be used by Dynatrace to connect to Salesforce, must be placed in the Activegate filesystem**

Creating a Connected App

In Salesforce Lightning go to `Setup`, then `Apps > App Manager`
Click `New Connected App`

blocked URL

Give the app a name, add the contact email
Under `API (Enable OAuth Settings)`, enable:

- `Enable OAuth Settings`
- Use digital signatures

The callback URL won't be used, you can use something like `http://localhost`
Upload the `cert.pem` file we've created under `Use digital signatures`

blocked URL

Under `Selected OAuth Scopes`, add:

- The `(api)` scope, that should be called `Manage user data via APIs (api)`
- The `(refresh_token, offline_access)` scope, that should be called `Perform requests at any time (refresh_token, offline_access)`

The names might differ depending on the version of Salesforce, but we are looking for the ones that end with `(api)` and `(refresh_token, offline_access)`

blocked URL

Leave all other options as they are, and hit `Save`

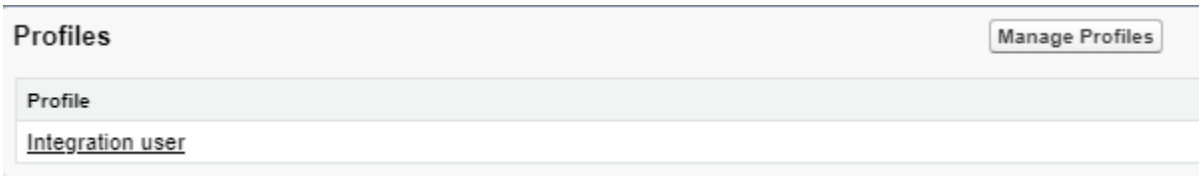
We need to setup the `OAuth Policy` permitted users now On the connected app page, hit `Manage`, then `Edit Policies`

Under `OAuth Policies` select `Admin approved users are pre-authorized`

blocked URL

Hit save.

On the same connected app page, under `Profiles`, click `Manage Profiles` Add a profile for users that are approved to use this connected app.



Troubleshooting

Troubleshooting

The logs under `%PROGRAMDATA%` (windows) or `/var/lib` (Linux) give us more details in case of failures

The full path is `/var/lib/dynatrace/remotepluginmodule/log/remoteplugin/custom.remote.python.salesforce_eventstream/SalesforceEventStream.log`

A good log:

blocked URL

Errors will also be sent to a Custom Device, example for an error when using the lightning URL instead of the classic URL:

blocked URL

Steps that would performed in Dynatrace is updated in this Confluence documentation https://btsconfluence.abbvie.com/x/XwulCw