Micro Focus Security
ArcSight FlexConnector for REST

FlexConnector REST

Developer's Guide

Document Release Date: April 30, 2021
Software Release Date: April 30, 2021

Legal Notices

Copyright Notice

Trademark Notices

Support

Contact Information

Phone	A list of phone numbers is available on the Technical Support Page: https://softwaresupport.softwaregrp.com/support-contact-information
Support Web Site	https://softwaresupport.softwaregrp.com/
ArcSight Product Documentation	https://community.softwaregrp.com/t5/ArcSight-Product-Documentation/ct-p/productdocs

Revision History

Date	Description
04/30/2021	Updated the troubleshooting section.
07/07/2020	Added steps on how to enable SNI manually.
09/19/2019	First edition of this Configuration Guide, for initial support of these events.

Contents

Chapter 1: ArcSight FlexConnector for REST

The FlexConnector for REST provides a configurable method to collect security events from cloud-based applications such as Salesforce, Google Apps, and so on.

REST FlexConnector Overview

The connector framework lets you develop FlexConnectors to collect events by configuring:

- OAuth2 for authentication with the vendor. See "OAuth2 Authentication" on the next page.
- Basic authentication. See "Basic Authentication " below.
- REST API endpoints exposed by the vendor for event collection. See "REST API Endpoints" on the next page.
- JSON parsers for parsing and mapping data (retrieved from the REST APIs). See "JSON Parsers" on page 8.

Before You Begin

Before beginning to develop your FlexConnector:

1. To fit the profile for the REST FlexConnector technology, make sure the vendor supports:
   - Basic or OAuth2 Authentication
   - REST API endpoints
   - JSON data format
2. Decide what kind of data you want to collect and determine the REST API endpoint for that data. Consult vendor documentation to determine data availability.

To develop your connector, you should be familiar with FlexConnector development See the *FlexConnector Developer's Guide* for details.

> **Note:** Consult the vendor documentation to verify the supported browsers and browser versions. If a supported browser is not used, the vendor login page might not display properly, preventing login, and you will be unable to configure the connector.

Basic Authentication

You can configure your connector to use the Basic authentication method to obtain permission to collect events from the cloud application. In this case, the client provides an identifier and a shared secret (such as a user name and a password). Basic authentication is defined by RFC

2617. If the vendor’s website uses HTTPS as the protocol, then all communication will be secure and encrypted.

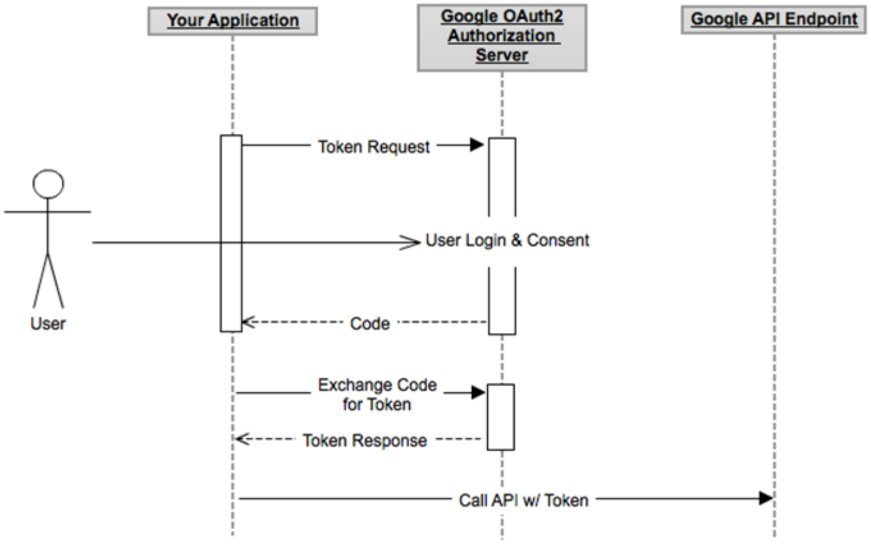
OAuth2 Authentication

You can configure your connector to use OAuth2 authentication to get permission for the connector to collect events from cloud applications. See "Registering Your Connector Application" on page 9 for details.

OAuth2 authentication ensures that the connector does not store a user’s login credentials (user name and password) for a cloud vendor application.

The connector redirects the user to the vendor's login page. When the user logs in and gives the permission to the connector, the vendor provides tokens to the connector to use to make API calls.

The following figure is an example of this authentication:



REST API Endpoints

Security events can be retrieved from the vendor’s predefined endpoints through HTTP requests. The REST FlexConnector supports data collection from cloud vendors where the retrieval of security events is exposed through REST-based APIs. Although events collected through REST APIs can be in XML or JSON formats, the REST FlexConnector supports only the JSON format. See "Determining the Events URL to Use" on page 13 for details.

For example, a REST API endpoint can look like this:

```
https://abc.com/events?created_after=<>&maxEvents=<>...
```

For the REST FlexConnector, the default query interval for the REST APIs is 30 seconds. This interval can be adjusted. See " Modifying Parameters to Optimize Connector Performance" on page 29.

JSON Parsers

The REST FlexConnector requires a JSON parser file. Most cloud vendors send data in JSON format. JSON supports these data types: arrays, objects, string, number, Boolean, or null. See "Create a JSON Parser File" on page 16 for details.

Prerequisites

- An Internet connection is required.
- Proxy information that might be required for the HTTP/HTTPS connection.
- Information required for OAuth2 authentication. See "Creating OAuth2 Client Properties File" on page 11 for details.
- The events URL, which is used as an endpoint for retrieving the events from the vendor. See "Determining the Events URL to Use" on page 13 for details.

Next Steps

Follow the steps in "Developing the FlexConnector " on page 9 to prepare for the install and configuration of the REST FlexConnector. After that, see "Installing and Configuring the REST FlexConnector" on page 22 to run the Configuration Wizard to install and configure the connector.

Chapter 2: Developing the FlexConnector

To develop a REST FlexConnector, perform the tasks described in this chapter, and then run the connector Configuration Wizard (described in "Preparing to Install the FlexConnector" on page 22). See "Before You Begin" on page 6 before starting your configuration tasks.

Registering Your Connector Application

For OAuth2 authentication, contact the vendor and register the connector application. When you register, the vendor will supply values for creating a vendor-specific `OAuth2 Client Properties` file. The content and format of the OAuth2 Client Properties file are discussed in "Creating OAuth2 Client Properties File" on page 11.

> **Note:** You must also supply the vendor with a `redirect_uri` value. The `redirect_uri` must be registered with the vendor before you attempt to configure the connector or the authentication will fail.

For Basic authentication, contact the vendor to obtain a user name and password along with the events URL.

The following sections provide details on registering a connector application with the vendors using OAuth2 authentication. The instructions do not apply to vendors using Basic authentication. You can register with vendors other than Salesforce and Google Apps, but the `auth_url`, `token_url`, and `scope` values must be obtained from the vendor's documentation. The values for Salesforce and Google Apps are supplied in the following sections.

Salesforce OAuth2 Registration and Values

To register your Salesforce connector application:

1. Follow this link to register the application: https://salesforce.com.
2. Login to Salesforce and go to **App Setup > Create > Apps > New**.
3. Specify a name.
4. Set the **CallbackURL** (`redirect_uri`) as `https://localhost:<port>/<path>`
5. Select **Scope > Full Access**.
6. After the application is created, write down the **Consumer Key** (`client_id`) and **Consumer Secret** (`client_secret`) as these will be needed later during connector configuration.

For details, see the Salesforce developer's documentation, which is available at: http://wiki.developerforce.com/page/Digging_Deeper_into_OAuth_2.0_on_Force.com

Salesforce OAuth2 values

After you register, create a Salesforce vendor-specific OAuth2 Client Properties file using these values (follow the steps for file creation described in "Creating OAuth2 Client Properties File" on the next page):

- client_id=<your client id>
- client_secret=<your client secret>
- auth_url=https://login.salesforce.com/services/oauth2/authorize
- token_url=https://login.salesforce.com/services/oauth2/token
- redirect_uri=https://localhost:<port>/<path>

> **Note:** Verify with the vendor that the URLs provided above are correct. These may change, so be sure to have the most current URLs from the vendor.

Google Apps OAuth2 Registration and Values

To register your Google Apps connector application:

1. Follow this link to register the application: https://code.google.com/apis/console/
2. Click **Create A Project**.
3. Click **Services > Enable Audit API**.
4. Accept the license terms.
5. Click **API Access > Create an OAuth2 Client ID.**
6. Choose **Web Application**.
7. Enter the `redirect_uri` as `https://localhost:<port>/<path>`.
8. Write down the `client_id` and `client_secret` as these will be needed later during connector configuration.

For details, see the Google Apps developer’s documentation, which is available at: https://developers.google.com/accounts/docs/OAuth2

Google Apps OAuth2 values

After you register, create a Google Apps vendor-specific OAuth2 Client Properties file using these values (follow the steps for file creation described in "Creating OAuth2 Client Properties File" on the next page):

- client_id=<your client id>
- client_secret=<your client secret>
- auth_url=https://accounts.google.com/o/oauth2/auth

- token_url=https://accounts.google.com/o/oauth2/token
- redirect_uri=https://localhost:<port>/<path>
- scope=https://apps-apis.google.com/a/feeds/policies/ https://www.googleapis.com/auth/apps/reporting/audit.readonly

> **Note:** Verify with the vendor that the URLs provided above are correct. These may change, so be sure to have the most current URLs from the vendor.

Creating OAuth2 Client Properties File

When using OAuth2 authentication, create an OAuth2 Client Properties file for each vendor from which you want to collect events. You can name the file to contain the vendor's name to help you keep track of your properties files. For example, an OAuth2 Client Properties file for the vendor Google could be named `googleclient.properties`. The file can reside on your local drive. You later browse for this file to add it to the "Enter parameter details" window in the connector Configuration Wizard.

This is the template for the OAuth2 Client Properties file:

> **Note:**
> - The OAuth2 Client Properties file must begin with a blank line or a comment statement.
> - The order of the properties is important. For example, if you place `auth_url` before `client id`, then connector will respond with an error.

```
# comment statement
client_id=<your client id>
client_secret=<your client secret>
redirect_uri=https://localhost:<port-number>/<path>
auth_url=<available from cloud service provider>
token_url=<available from cloud service provider>
scope=<value, if any

timestamp_format_of_api_vendor=<null or <default> or timestamp format vendor
support for API>
```

> **Note:** If you want to retrieve events from Salesforce or Google Apps, see "Salesforce OAuth2 values" on the previous page or "Google Apps OAuth2 values" on the previous page for details on the values to use in vendor-specific OAuth2 Client Properties files.

The parameters and expected values in the OAuth2 Client Properties file are described in the following table.

Parameter	Description
client_id	This value is provided by the vendor when you register an application.
client_secret	This value is also provided by the vendor when you register an application. This value is obfuscated. The values `client_id` and `client_secret` helps the vendor identify an application.

redirect_uri	You configure this when you register an application. This is the URL to which the vendor sends the authorization code.
	For the REST Flex Connector, the `redirect_uri` must be on the local host. Both http and https schemes are supported. This URL should be of the form `https://localhost:<port>/<path>`. Examples:
	`http://localhost:8081/oauth2callback`
	`https://localhost:8081/oauth2callback`
	The `<port>` in this URL can be configured to any available port.
	Specify this URL when you register the application with the vendor. The connector allows either http or https, and the URL must be redirected to the unused port, `<port>` of local host `<localhost>` so that the authorization code can be captured automatically after vendor authentication. For an HTTPS connection, it shares the connector's default self-signed certificate, `remote-management.p12` located in the `user/agent` directory.

auth_url	This is the URL of the vendor to which the initial request must be made to get an authorization code. Consult the vendor documentation to get this URL.
token_url	This is the URL of the vendor to which the request for an Access Token will be made. Consult the vendor documentation to get this URL.
scope	Required. Specifying a value for the `scope` parameter is optional, but the parameter itself is not and must appear in your configuration. The `scope` parameter allows applications to inform you and the vendor what type of information is to be retrieved from the vendor on behalf of the user. If there is more than one `scope`, you can specify these as a space-separated list of values.
timestamp_ format_of_api_ vendor	Required. Time format can be null or "default". The time is taken as millisecond. Ex: 1558502432847. It can also directly transmit the time format that the provider supports for the API. Example: timestamp_format_of_api_vendor =yyyy-MM-dd'T'HH:mm:ss.SSS'Z'. timestamp_format_of_api_vendor = To see timestamp formats, please refer to https://help.sumologic.com/03Send-Data/Sources/04Reference-Information-for-Sources/Timestamps%2C-Time-Zones%2C-Time-Ranges%2C-and-Date-Formats

Notes

- The access token is initially obtained during the connector configuration and will be used during event retrieval while the connector is running. Because OAuth2 gives an application temporary access permission, the access token will expire after a period of time and must be refreshed.
- After successful authentication, all the OAuth2 Client Properties, as well as access token and refresh token are persisted in the `agent.properties` file. Tokens and secrets are obfuscated.

Determining the Events URL to Use

For both Basic and OAuth2 authentication types, the events URL is the REST API endpoint used by the connector to get events from a vendor. A general discussion of REST endpoints, and details on using REST endpoints from Salesforce and Google Apps, follows.

Refer to the vendor documentation to determine which REST URL will provide the events you want. Also, be sure that the user who configures the connector has the privileges to access that URL. If you need a different events URL from those vendors, or from a vendor other than those mentioned, see "About the REST FlexConnector Configuration Tool (restutil)" on page 34.

General Information about REST API Endpoints

An events URL has query parameters in the path so that the user or the application can pass the value to retrieve the events that meet the specified condition.

- **Querying Based On TimeStamp:** Event retrieval can be limited by passing the start time. An example of an events URL:

  ```
  https://abc.com/events?start_time=<time>
  ```

  The connector relies on a timestamp query parameter for the start time (in the above example, it is `start_time`). To identify such a timestamp query parameter in the events URL, the connector expects to see a `$START_AT_TIME` after the `start_time` parameter. So, the full events URL in this example is:

  ```
  https://abc.com/events?start_time=$START_AT_TIME
  ```

  A tag and a start timestamp parameter are needed to collect the latest events and prevent duplicate events.

  In addition to specifying the place holder `$START_AT_TIME`, you also must correctly map the `event.deviceReceiptTime` in the parser to the time at which the event happened (as reported by the device).

- **Rate Limiting:** REST API endpoints can also support querying for a maximum number of events. Google, for example, has the query parameter `limit` to use for this purpose. An example of a REST API is:

  ```
  https://api.google.com/2.0/events?stream_type=admin_logs&created_
  after=$START_AT_TIME&limit=500
  ```

  In the above example, a maximum of 500 events is requested from Google. If there are more than 500 events, then Google expects further API calls to retrieve the remaining events. Some vendors (such as Google Apps and Salesforce) provide **Next URLs** in the JSON response to let clients retrieve the remaining events. The connector supports this by making requests to the **Next URLs**. For further details, see "Create a JSON Parser File" on page 16. Other vendors do not provide such URLs and the client will have to query more times to get the latest events.

There can be any number of other query parameters in a REST endpoint. All of them can be provided in the events URL. For example, an API could look like this:

```
https://abc.com/events?start_time=$START_AT_
TIME&max=100&param1=PARAM1&param2=PARAM2&param3=PARAM3...
```

Some vendors may have dynamic content in their URLs. For example, Google Apps Events URL should have a `CustomerID` (that is dependent on the user who logged in) and, therefore, is not static. The connector expects the URL to be static. To obtain the dynamic content of a URL, refer to the vendor documentation and execute the REST Flex Configuration Tool to get the relevant content. Once you ensure that you have the entire URL that the connector can execute, you can run the connector.

The following subsections discuss three well-known vendors (Salesforce and Google Apps) and how to use their event URLs.

> **Note:** If the APIs discussed in the following sections could be changed by the vendor, refer to the vendor's documentation for the correct APIs.

Salesforce REST APIs

Salesforce exposes many public APIs. All these APIs can support Salesforce Query Language (SOQL) queries. For example, an API to retrieve Login History data from Salesforce is:

```
https://<INSTANCE>.salesforce.com/services/data/v23.0/query/?q=SELECT
LoginTime,LoginType,UserId,Status,SourceIp,LoginUrl From LoginHistory WHERE
LoginTime>$START_AT_TIME
```

In the above URL, `<INSTANCE>` is the instance name, which differs for different users. You can find your `<INSTANCE>` by logging into Salesforce. After you log in successfully, you will see your Home Page on Salesforce. The URL of this page can appear like this:

```
https://na14.salesforce.com/<value>/<value>/...
```

In this case, the `<INSTANCE>` value is na14. Once you find your instance name, replace `<INSTANCE>` in the events URL so that the final URL for example, is:

```
https://na14.salesforce.com/services/data/v23.0/query/?q=SELECT
LoginTime,LoginType,UserId,Status,SourceIp,LoginUrl From LoginHistory WHERE
LoginTime>$START_AT_TIME
```

To use this URL, copy-and-paste it into the Events URL field in the **Enter the parameter details** window in the connector Configuration Wizard. After completion of the connector installation process, the `agents.eventsurl=` value can be modified in the `$ARCSIGHT_HOME\current\user\agent\agent.properties` file.

The query part in the above example is:

```
SELECT LoginTime,LoginType,UserId,Status,SourceIp,LoginUrl From LoginHistory
WHERE LoginTime>$START_AT_TIME
```

The `$START_AT_TIME` is used by the connector to identify the timestamp parameter. In the example above, the timestamp parameter is `LoginTime`. You can specify any other valid SOQL query as part of the events URL. The following is a general SOQL example:

```
SELECT A, B, C FROM D WHERE B > $START_AT_TIME
```

In this second example, B should be the timestamp parameter.

For further details, see the *Salesforce REST API Developer's Guide*, available at: http://www.salesforce.com/us/developer/docs/api_rest/index.htm.

Google Apps REST API

Google Apps exposes an Admin Audit API that can be used to monitor a Google Apps account. This API is:

```
https://www.googleapis.com/apps/reporting/audit/v1/<CustomerID>/207535951991?
maxResults=50&startTime=$START_AT_TIME
```

The `<CustomerID>` portion of this URL depends on the Google Apps account to be monitored. Before you start the connector Configuration Wizard, get the `<CustomerID>`, replace the`<CustomerID>` parameter in the events URL shown above with the actual value, and paste it into the **Events URL** field in the **Enter the parameter details** window during connector configuration. After completion of the connector installation process, the `agents.eventsurl=` value can be modified in the `$ARCSIGHT_HOME\current\user\agent\agent.properties` file.

To get the `<CustomerID>` value, use the REST FlexConnector Configuration Tool. See "About the REST FlexConnector Configuration Tool (restutil)" on page 34 for details.

Also, ensure that the Provisioning API is enabled on the Google Apps account you are about to monitor. To do this, login to the Google Apps account, go to and select **Control Panel > Domain Settings > User Settings > Enable Provisioning API**.

For further details, see the Google Developer’s Guide, available at: https://developers.google.com/google-apps/admin-audit/get_started.

Run restutil to Obtain a Refresh Token for ArcSight Management Center

Before configuring the connector on the Connector Appliance/ArcSight Management Center, obtain a refresh token, which the connector will use to access the vendor's log data.

Use the REST FlexConnector Configuration Tool (restutil) to obtain a refresh token. See "About the REST FlexConnector Configuration Tool (restutil)".

1. Ensure the connector software is installed on a host machine that has access to a web browser. See "Installing Core Software" for more information.
2. After installing the connector core software, navigate to `$ARCSIGHT_HOME\current\bin`.
3. To retrieve a refresh token, invoke the tool with the token command:

   ```
   arcsight restutil token [-proxy <proxy-info>] -config <OAuth2 Client
   Properties File>
   ```

   For example:

   ```
   arcsight restutil token -proxy proxy.atlanta.mf.com:8080 -config
   c:\temp\google.properties
   ```
4. A web browser opens and prompts you to log into the vendor application. Enter your user name and password and click through to access the vendor application.
5. The refresh token string is displayed in the command line window. Copy this string into the **Refresh Token** field during connector configuration.

Create a JSON Parser File

A JSON parser file must be created prior to configuring the connector. The parser file name will be required during connector configuration.

At this time, you will provide only the prefix of the parser file name. For example, if you provide the configuration file name as “google”, the connector expects the file `google.jsonparser.properties` file in `$ArcSight_Home/user/agent/flexagent` directory. If the connector cannot load this file, the connector configuration does not succeed, and the connector will not be able to parse the JSON data. You can learn more about JSON from http://www.json.org/ and http://en.wikipedia.org/wiki/JSON.

Example JSON Structure

The following is an example of JSON structure for the REST FlexConnector to help you create your own JSON parser:

```
{
    "kind": "audit#activities",
    "items": [
        {
            "kind": "audit#activity",
            "id": {
                "time": "2013-01-25T00:49:29.123Z",
                "uniqQualifier": "-6297847723024543031",
                "applicationId": "207535951991",
                "customerId": "ABCD1234"
            },

            "actor": {
                "callerType": "USER",
                "email": "john@abc.com"
            },

            "ownerDomain": "abc.com",
            "ipAddress": "1.1.1.1",
            "events": [

                {
                    "eventType": "USER_SETTINGS",
                    "name": "CREATE_USER",
                    "parameters": [

                        {
                            "name": "USER_EMAIL",
                            "value": "doe@abc.com"
                        }

                    "additional" : {
                            "additional1" : "add"
                                }
                    ]
                }
            ]
        }
    ],
    "next":
"https://www.googleapis.com/apps/reporting/audit/v1/ABCD1234/207535951991?max
Results=10&alt=json&continuationToken=A:1357594673492000:-
6925963958411121628:207535951991:C02deea2k"
}
```

Example JSON Parser

An example JSON parser for such a structure would be:

```
trigger.node.location=/items/events

token.count=10

        token[0].name=kind
        token[0].type=String
        token[0].location=/kind

        token[1].name=kindOfItem
        token[1].type=String
        token[1].location=../../kind

        token[2].name=additional
        token[2].type=String
        token[2].location=additional/additional1

        token[3].name=time
        token[3].type=String
        token[3].location=../../id/time

        token[4].name=nextUrl
        token[4].type=String
        token[4].location=/next

        ...
        ...
        ...

        (End Of Token Definitions)

        event.deviceReceiptTime=__createOptionalTimeStampFromString
(time,"yyyy-MM-ddTHH:mm:ss.SSSZ")
        event.externalId=uniqQualifier
        event.deviceCustomString6=nextUrl
        ...
        ...
```

The syntax of a JSON parser is similar to that of XML parser. The parser consists of `trigger.node.location`, token properties, and event mappings:

trigger.node.location

This is mandatory. It is used to specify the node or nodes in a JSON parser for which events are to be built. It can be the root (`trigger.node.location = /`) or any other node specified relative to the root (`trigger.node.location=/items/events`) in the JSON. If the node specified as the `trigger.node.location` is an array, an event is built for every element in the array. If the node is not an array, one event is built corresponding to the object specified in the `trigger.node.location`. If the `trigger.node.location` has arrays in its path, then the number of events generated is the product of the number of elements in all arrays in the path.

In "Example JSON Parser" on the previous page, since the `trigger.node.location` is `/items/events` and both `items` and `events` are arrays, the number of events generated = (Number of elements in `items`) * (Number of elements in `events`). If the location were `items`, the number of events generated would only be the number of elements in `items`.

Token Properties

- `token.name`—A name you give to the token. Use this name later when you assign the token to a event schema field.
- `token.type`—The data type of the token.
- `token.location`—The location of the token in the JSON. Specify the location of the token in one of these ways:
  - Relative to the root. For example, `token[0]` in the example has the location `/kind` which has been specified relative to the root /
  - Relative to the `trigger` node. For example, `token[1]` in the JSON parser example has the location `../../kind`, which has been specified relative to the trigger node. (Which in this example is every element of the `events` array).
- You can specify the `token.location` by going up or down from a trigger node. For example, in `../../kind`, by specifying the first `..`, you are going to the array `events` of which the trigger is an element. By specifying the second `..`, you are going to the array `items` of which the `events` is an element.
- You can also specify the `token.location` to a particular element of an array. For example, a `token.location` in the JSON parser example can be `parameters[1]`, if we want the token location to be the second element of the `parameters` array.

- The total number of tokens should match the `token.count`.
- The token concept is the same as in XML, Regex, Syslog parsers, except for the `token.location` concept.

Event Mappings

Event mappings are the same as for any parser type.

If you are using the `$START_AT_TIME` in the REST API Events URL, you should map the timestamp at which the event happened to `event.deviceReceiptTime.` See "Determining the Events URL to Use" on page 13 for details. In the JSON parser example, the token `time` is mapped in that way. Most of the timestamps reported on the Web are in ISO-8601 format. To parse such timestamps, you can use the parser operation: `__createOptionalTimeStampFromString`, and specify the timestamp format as: `YYYY-MM-DDThh:mm:ss.SSSX`.

If the REST API supports sending next URLs (`nextURLs`) in the JSON to support rate limiting, then map the next URL to `event.deviceCustomString6,` as in the example in "Example JSON Parser" on page 18.

If you want to use the **next URL** feature to enable caching, then map a field that is unique for every event to `event.externalID`.

Viewing the Raw JSON Data

There are two methods to view raw JSON data:

- See the vendor's documentation, which will contain the information on the JSON structures sent by the REST APIs.
- Enable debug on the connector to see the JSON data retrieved by the connector:

  a. Add these properties to the `$ARCSIGHT_HOME\current\user\agent\agent.properties` file:

  ```
  log.global.debug=true
  log.channel.file.property.package.com.arcsight=0
  ```

  b. Save the `agent.properties` file.

  c. Restart the connector.

  d. Raw JSON data is then available in `$ARCSIGHT_HOME\current\logs\agent.log.`

Next Steps

When you have completed the configuration steps described in the previous sections of this chapter, you are ready to run the connector Configuration Wizard. The Wizard lets the user install the REST FlexConnector, enter the configuration parameters, and authenticate with the vendor. See "Installing and Configuring the REST FlexConnector" on page 22.

Chapter 3: Installing and Configuring the REST FlexConnector

This chapter describes the steps to install and configure the REST FlexConnector.

Preparing to Install the FlexConnector

Before you install the FlexConnector, make sure that the ArcSight products with which the connector will communicate have already been installed correctly (such as ArcSight ESM or ArcSight Logger). This configuration guide takes you through the installation process with ArcSight Manager (encrypted) as the destination.

For complete product information, read the Administrator's Guide as well as the Installation and Configuration guide for your ArcSight product.

Before installing the FlexConnector, be sure the following are available:

- Local access to the machine where the FlexConnector is to be installed
- Vendor login credentials (user name and password). During the configuration, you are redirected to the vendor’s login page, where you will log into the vendor’s application using your vendor credentials. After you log into the vendor application, the connector can access and collect vendor log data.

Unless specified otherwise at the beginning of this guide, this connector can be installed on all ArcSight supported platforms; for the complete list, see the *SmartConnector Platform Support* document, available from the Micro Focus SSO and Protect 724 sites.

> **Note:** On the Linux platform, if you are logged in as `root` and use Firefox, some versions of the browser can cause the browser launched by the connector during configuration not to open. If you see this issue, try configuring the connector as a non-root user. If you configure the connector as a non-root user, however, you cannot run the connector as a service.

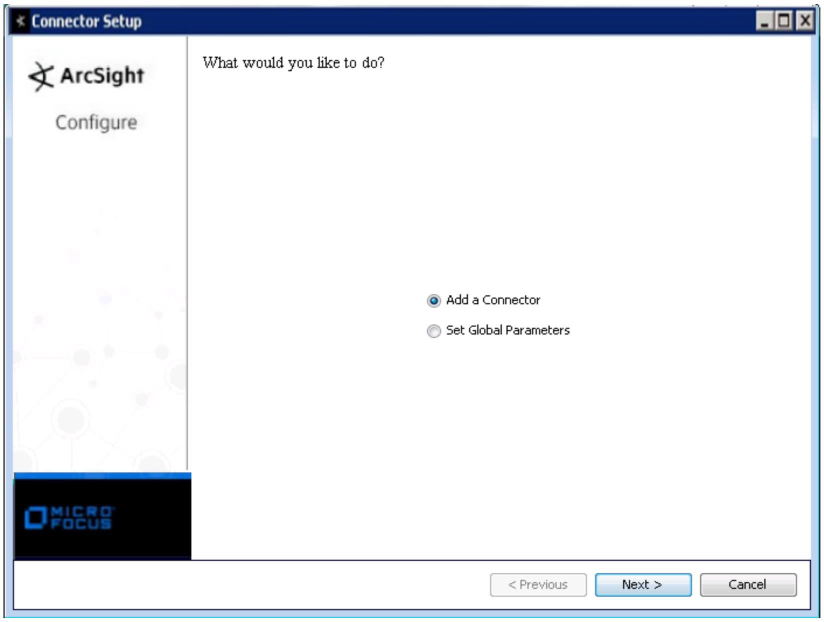
Installing Core Software

1. Download the SmartConnector executable for your operating system from the Micro Focus SSO site. (FlexConnectors are a type of SmartConnector.)
2. Start the SmartConnector Installer by running the executable.

   Follow the installation wizard through the following folder selection tasks and installation of the core connector software:

Introduction
Choose Install Folder
Choose Shortcut Folder
Pre-Installation Summary
Installing...

3. When the installation of connector core component software has completed, the following window is displayed.



What would you like to do?

● Add a Connector
○ Set Global Parameters

Adding JSON Parser

1. Before configuring the connector, you must exit the wizard to make your JSON parser available to the connector. Click **Cancel** to exit the wizard.
2. Copy your JSON parser file into the `$ARCSIGHT_HOME\current\user\agent\flexagent` directory. See "Create a JSON Parser File" for details on creating the JSON parser file.
3. Execute `runagentsetup` from `$ARCSIGHT_HOME\current\bin` to return to the wizard.

Continue with "Setting Global Parameters (Optional)" if you want to set FIPS mode, remote management, or preferred IP version. Otherwise, continue with "Configuring Connector Parameters."

Setting Global Parameters (Optional)

If you choose to perform any of the operations shown in the following table, do so before adding your connector. After installing core software, you can set the following parameters:

Global Parameter	Setting
FIPS mode	Set to **Enabled** to enable FIPS compliant mode. To enable FIPS Suite B Mode, see the *SmartConnector User Guide* under "Modifying Connector Parameters" for instructions. Initially, this value is set to **Disabled**.
Remote Management	Set to **Enabled** to enable remote management from [[[Undefined variable _ ARST_Variables.Management Center]]]. When queried by the remote management device, the values you specify here for enabling remote management and the port number will be used. Initially, this value is set to **Disabled**.
Remote Management Listener Port	The remote management device will listen to the port specified in this field. The default port number is 9001.
Preferred IP Version	When both **IPv4** and **IPv6** IP addresses are available for the local host (the machine on which the connector is installed), you can choose which version is preferred. Otherwise, you will see only one selection. The initial setting is **IPv4**.

The following parameters should be configured only if you are using Micro Focus SecureData solutions to provide encryption. See the *Micro Focus SecureData Architecture Guide* for more information.

Global Parameter	Setting
Format Preserving Encryption	Data leaving the connector machine to a specified destination can be encrypted by selecting ‘Enabled’ to encrypt the fields identified in ‘Event Fields to Encrypt before forwarding events. If encryption is enabled, it cannot be disabled. Changing any of the encryption parameters again will require a fresh installation of the connector.
Format Preserving Host URL	Enter the URL where the Micro Focus SecureData server is installed.
Proxy Server (https)	Enter the proxy host for https connection if any proxy is enabled for this machine.
Proxy Port	Enter the proxy port for https connection if any proxy is enabled for this machine.

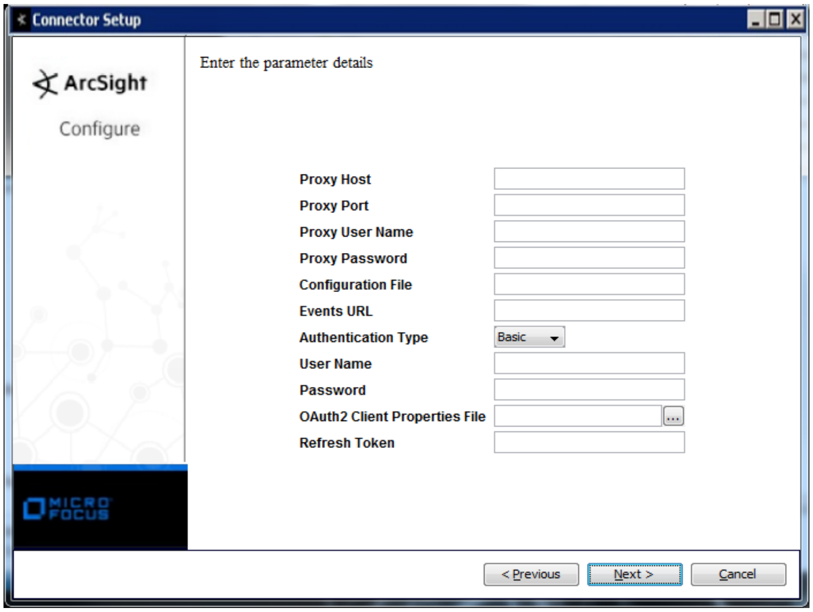
Global Parameter	Setting
Format Preserving Identity	The Micro Focus SecureData client software allows client applications to protect and access data based on key names. This key name is referred to as the identity. Enter the user identity configured for Micro Focus SecureData.
Format Preserving Secret	Enter the secret configured for Micro Focus SecureData to use for authentication.
Event Fields to Encrypt	Recommended fields for encryption are listed; delete any fields you do not want encrypted from the list, and add any string or numeric fields you wish to be encrypted. Encrypting more fields can affect performance, with 20 fields being the maximum recommended. Also, because encryption changes the value, rules or categorization could also be affected. Once encryption is enabled, the list of event fields cannot be edited.

After making your selections, click **Next**. A summary screen is displayed. Review the summary of your selections and click **Next**. Click **Continue** to return to the "Add a Connector" window. Continue the installation procedure with "Configure Connector Parameters".

Configuring Connector Parameters

Configure Connector Parameters

1. Select **Add a Connector** and click **Next**.
2. Select **ArcSight FlexConnector REST** and click **Next**.
3. Enter the required parameters to configure the connector, then click **Next**.



Enter the parameter details

Proxy Host
Proxy Port
Proxy User Name
Proxy Password
Configuration File
Events URL
Authentication Type: Basic
User Name
Password
OAuth2 Client Properties File ...
Refresh Token

Note: If you do not use a proxy to access the Internet, leave the proxy fields blank and enter the other parameter values.

Parameter	Description
Proxy Host	Enter the proxy host IP address or name. This value is required if you use a proxy to access the Internet.
Proxy Port	Enter the proxy port. This value is required if you use a proxy to access the Internet.
Proxy User Name	Enter the proxy user name. This value is optional for additional proxy authentication. If you specify a proxy user name, you must also specify a proxy password.
Proxy Password	Enter the password for the proxy user specified in the Proxy User Name field. This value is optional for additional proxy authentication. This field is required only if you have specified a proxy user name.
Configuration File	Enter the name of the parser file once you have ensured the parser file has been copied into the `$ARCSIGHT_HOME\current\user\agent\flexagent` directory. For example, for `$ARCSIGHT_HOME\current\user\agent\flexagent\google.jsonparser.properties`. You can enter the prefix `google`, and the connector assumes the file is named `google.jsonparser.properties` and resides in `$ARCSIGHT_HOME\current\user\agent\flexagent`. See "Create a JSON Parser File" on page 16 for details on parser creation.
Events URL	Enter the events URL. This is the REST API endpoint used by the connector to retrieve the events. See "Determining the Events URL to Use" on page 13 for general information about REST API endpoints, and specific information about the REST API endpoints for the vendors Salesforce and Google Apps.
Authentication Type	Select Basic or OAuth2 as the type of authentication to be used.
User Name	For Basic authentication, enter the User Name.
Password	For Basic authentication, enter the Password.
OAuth2 Client Properties File	For OAuth2 authentication browse for the OAuth2 Client Properties File. You should have created this file from values you obtained when you registered your connector application, and acquired a redirect_uri. See "Registering Your Connector Application" on page 9 and "Creating OAuth2 Client Properties File" on page 11 for details. Create a unique OAuth2 Client Properties File for each vendor from which you will collect events.
Refresh Token	For OAuth2 only. Enter the refresh token. For information on how to produce the Refresh Token, see "Run restutil to Obtain a Refresh Token for ArcSight Management Center" on page 16. (This applies only to users running the FlexConnector in the Connector Appliance or ArcSight Management Center environment. Other users, leave this field blank.)

4. This step applies to the OAuth2 authentication. A browser window will be launched by the connector to let you to log in to the vendor application. Be sure to log in using the browser window launched by the connector, and do not use any other browser window for your vendor application log in. You do not need to manually open the browser to log into the vendor application.

   If there are multiple windows open for the vendor application in your browser, verify that you log in using the window that was opened by the connector wizard. You might need to close other browser windows you have open for the vendor application to ensure that you are logging in to the correct window.

   To log in, enter your vendor application user name and password and click **Log In**. After logging in, another page could be displayed, asking you to give permission to your OAuth2 Client. This permission is required for the connector wizard to perform authentication. After you give the permission, if the `redirect_uri` is an https URL, you are redirected to a URL on the local host, and you view a page requesting that you trust the certificate provided by the connector running on the local host. You must trust this certificate. Note that this trust is not required if you specify a http URL for the `redirect_uri`.

   When the connector launches a browser window, it attempts to use the default browser configured for your system. If the default browser does not launch, it will try to launch using other browsers (Firefox, Google Chrome, Internet Explorer, Konqueror, or Mozilla). Verify that you have one of these browsers configured on your system. Also, ensure that the proxy settings for your browsers are configured correctly so that you can access the Internet through your browser. The connector configuration wizard waits for 3 minutes for you to log in and grant permission.

   If you have not logged in after 3 minutes, a warning message is displayed indicating that the connector parameters did not pass a security verification. In such cases, users can:

   a. **Refresh Token**

      i. Click **Yes** to continue with the installation without logging in. Note that when the connector starts retrieving events, the parameter **Refresh Token** must be modified and a login is required.

      ii. Or, you obtain the refresh token from the command line, and enter the Refresh Token. For more information, see "Run restutil to Obtain a Refresh Token for ArcSight Management Center" on page 16

      > **Note**: After obtaining the token refresh, the value can be changed from the agent.properties file. However, this is not recommended, as the refresh values are not encrypted.

b. **Modify the Refresh Token Parameter**

   i. Open command line %ARCSIGHT_HOME%/bin

   ii. Run: arcsight agentsetup –c

      A. Select **Modify Connector** and continue,

      B. Choose **Modify connector parameters** and continue.

      C. From the textbox, enter the Refresh Token obtained, for more information see "Run restutil to Obtain a Refresh Token for ArcSight Management Center" on page 16 .

      D. Click No and continue.

      A login request pops to refresh token and the system automatically populates the textbox **Refresh Token**.

After you log into the vendor application, continue connector configuration. The next page is displayed automatically. To continue, be sure to return to this ArcSight Connector Setup window. You might have to look under the vendor application window to find the Connector Setup window.

Selecting a Destination and Completing Installation

1. Make sure **ArcSight Manager (encrypted)** is selected and click **Next**. For information about this and other destinations listed, see the *ArcSight SmartConnector User Guide* and the Administrator's Guide for your ArcSight product.
2. Enter the **Manager Host Name**, **Manager Port**, and a valid ArcSight **User Name** and **Password**. This is the same user name and password you created during the ArcSight Manager installation. Click **Next**.
3. Enter a name for the connector and provide other information identifying the connector's use in your environment. Click **Next**; the connector starts the registration process.
4. The certificate import window for the ESM Manager is displayed. Select **Import the certificate to the connector from destination** and click **Next**. The certificate is imported and the **Add Connector Summary** window is displayed. If you select **Do not import the certificate to connector from destination**, then the connector installation will end.
5. Review the Add Connector Summary and click **Next**. If the summary is incorrect, click **Previous** to make changes.
6. The wizard now prompts you to choose whether you want to run the connector as a stand-alone process or as a service. If you choose to run the connector as a stand-alone process, skip step 7. If you choose to run the connector as a service, the wizard prompts you to define service parameters.

7. Enter values for **Service Internal Name** and **Service Display Name** and select **Yes** or **No** for **Start the service automatically**. The **Install Service Summary** window is displayed when you click **Next**.
8. Click **Next**.

To complete the installation, choose **Exit** and click **Next**.

> **Note:** Save any work on your computer or desktop and shut down any other running applications (including the ArcSight Console, if it is running), then shut down the system.

Complete any tasks needed in " Modifying Parameters to Optimize Connector Performance", then continue with the "Running the SmartConnector".

For connector upgrade or uninstall instructions, see the *SmartConnector User Guide*.

Modifying Parameters to Optimize Connector Performance

To optimize connector performance, you can modify parameter values to configure how frequently the connector retrieves events from the vendor. These parameters include `increasenexttimestampinmillis`, `maintaineventcache`, `queryfrequency`, `startattime`, `reauthenticate_onstartup`, and `useexpirytimetorefreshtoken`, which are described in step 2.

After SmartConnector installation, access the connector's parameters as follows:

1. From the `$ARCSIGHT_HOME\current\user\agent` directory open the file `agent.properties` in a pure ASCII text editor (such as Notepad++).
2. In the `agent.properties` file, locate the parameters whose values you want to modify.
   - The default value for the `increasenexttimestampinmillis` parameter is 1 millisecond. In most cases, the you will not have to change this, since most providers support milliseconds in their timestamps. If your provider does not support milliseconds, you must change this parameter to a second (1000 milliseconds) to prevent duplicate events. When you make this change, the next time the connector makes an API call, it starts 1 second after the last timestamp.
   - The default value for the `maintaineventcache` parameter is `false`. By default, no caching occurs. If you think you are getting duplicate events, you can enable caching by changing the value of `maintaineventcache` to `true`. Duplicates are suppressed based on the `event.externalId` (which is supposed to be unique) and `event.deviceReceiptTime.`

- The default value for the `queryfrequency` parameter is `30000 ms`; you can adjust this value to tune the connector performance. Note that `queryfrequency` influences the number of API calls the connector makes to the vendor. If vendor account has a limit for the number of API calls during a period of time, you can configure `queryfrequency` to reduce the number of API calls made by the connector. The greater the `queryfrequency` value, the fewer the number of API calls made by the connector over a period of time.

- Enter a value for the `startattime` parameter to specify an exact timestamp from which the connector is to start processing events.

  The format of this timestamp should be `yyyy-MM-dd’T’HH:mm:ss.SSSZ (Example: 2012-05-15T00:01:02.345-08:00)`. If no time is specified, all events will be processed.

- The default value for `reauthenticate_onstartup` is `false`. When you change this value to `true`, the connector will attempt to reauthenticate the user each time the connector starts. It will launch the browser and expect the user to log in and give permission to the OAuth2 Client application.

- The default value for `useexpirytimetorefreshtoken` is `true`. So, by default, the connector uses the expiry time of the access tokens (as sent by the vendor) to refresh the tokens, whenever needed. If the value is changed to `false`, the connector will ignore the expiry times sent by the vendor.

3. Save and exit the `agent.properties` file.
4. Restart the connector.

Running the SmartConnector

SmartConnectors can be installed and run in stand-alone mode, on Windows platforms as a Windows service, or on UNIX platforms as a UNIX daemon, depending upon the platform supported. On Windows platforms, SmartConnectors also can be run using shortcuts and optional Start menu entries.

If the connector is installed in stand-alone mode, it must be started manually and is not automatically active when a host is restarted. If installed as a service or daemon, the connector runs automatically when the host is restarted. For information about connectors running as services or daemons, see the *SmartConnector User Guide*.

To run all SmartConnectors installed in stand-alone mode on a particular host, open a command window, go to `$ARCSIGHT_HOME\current\bin` and run: **arcsight connectors**

To view the SmartConnector log, read the file `$ARCSIGHT_HOME\current\logs\agent.log`; to stop all SmartConnectors, enter **Ctrl+C** in the command window.

Server Name Indication

Server Name Indication (SNI) is a TLS extension, defied in RFC 4366. It enables TLS connections to virtual servers, in which multiple servers for different network names are hosted at a single underlying network address.

To enable SNI manually:

1. From

   `$ARCSIGHT_HOME/current/bin/scripts/jvmcommonparams.bat` (Windows)

   or

   `$ARCSIGHT_HOME/current/bin/scripts/jvmcommonparams.sh` (Unix)

   change the line

   `-Djsse.enableSNIExtension = false` to `-Djsse.enableSNIExtension = true`.
2. If the connector is being run as a service, from

   `user/agent/agent.wrapper.conf` change the line

   `-Djsse.enableSNIExtension=false` to `-Djsse.enableSNIExtension=true`.

Troubleshooting

The following problems can affect connector authentication and connection to the vendor.

- Redirect to the vendor login page does not occur during connector configuration on Linux. Some browsers do not retain the proxy settings for certain users, which can prevent the redirect to the vendor login page. If this occurs, log in as a user other than **root** and start the connector.
- If the proxy is not configured properly, the connection between the connector and the vendor will fail. This will cause the authentication to fail during the connector configuration.
- If a network failure occurs during connector configuration, the authentication will fail. However, event retrieval attempts will be made if the network failure occurs while the connector is running.
- Any invalid values in the OAuth2 Client Properties file will cause the authentication to fail.
- On the expiration of the access token (typically in an hour), the connector will automatically refresh the token using the refresh token sent by the vendor and continue event retrieval. However, if the refresh token expires, the access token cannot be refreshed and the event retrieval will be interrupted. In this case, the connector must be restarted.

Certificate Issue While Integrating the Flexconnector with Azure Sentinel Alerts.

The following certificate exception error may be displayed while configuring up the Flex Rest Api connector.

```
Error[1]: [Unable to use the Events URL.javax.net.ssl.SSLException:
Certificate for <sent-frc-api.azure-api.net> doesn't match any of the subject
alternative names: [*.azurewebsites.net, *.scm.azurewebsites.net, *.azure-
mobile.net, *.scm.azure-mobile.net, *.sso.azurewebsites.net]]
```

In the SmartConnector environment, the property to enable or disable Server Name Indication is disabled by default.

To enable the SNI:

- First, verify if the events URl is correct. If they are, based on your connector installation, choose one of the following steps:
- For stand alone installations, go to the `current\bin\script` folder and change to **True** the `-Djsse.enableSNIExtension` property in the following two files:

`connectors.bat`

`jvmcommonparams.bat`

- For service installations, go to the `current\user\agent` folder and change to **True** the `-Djsse.enableSNIExtension` property in the following file:

  `agent.wrapper.conf`

Salesforce API is Unable to Receive Events

The Salesforce API may not be able to receive events, and instead, the following error is displayed:

```
[2021-03-06 10:36:22,507][FATAL][com.arcsight.agent.loadable.agent._
FlexRestApiAgent] [refreshCredentials]There is no refresh token. Cannot
refresh the access token. Please reconfigure the connector to have the user
authenticated again.
```

The property `scope=full refresh_token` must be added.

To add the property `scope=full refresh_token`:

1. From the `agent.properties` file, go to `agents[0].reauthenticate_onstartup=false`.
2. Change the property to `true`.

Appendix A: About the REST FlexConnector Configuration Tool (restutil)

The `restutil` script lets you obtain the dynamic portion of an events URL. Some configuration values of the REST Flex Connector can include a dynamic portion that can be retrieved after running HTTP calls. For example, the Google Apps events URL has the customer id in the path and it can be retrieved via an authorized HTTP GET using the token obtained after the authentication is completed.

Use `restutil` to retrieve the dynamic portion of the events URL after installing core software and before beginning connector configuration. See "Installing Core Software" and "Configuring Connector Parameters".

restutil Configuration Tool Syntax

The following is the syntax of the `restutil` script.

```
arcsight restutil <command> <option-list>
<command> := [ authget | token | execute ]
<option-list> for authget := [ <proxy-option> ] <config-option> <url-option>
<option-list> for token := [ <proxy-option> ] <config-option>
<option-list> for execute := [ <proxy-option> ] <url-option> <execute-
options-list>
<execute-options-list> := <execute-options> [ <execute-options-list> ]
<execute-options> := [ <token-option> | <method-option> | <header-option> |
<query-option> | <content-option> ]
<proxy-option> := -proxy <proxy-info>
<proxy-info> := <host>:<port>[:<user>:<password>]
<config-option> := -config <properties-file>
<url-option> := -url "<url>"
<token-option> := -token <token>
<method-option> := -method [GET|PUT|POST|DELETE|HEAD]
<header-option> := -header "<parameter-list>"
<query-option> := -query "<parameter-list>"
<content-option> := -content "<parameter-list>"
<parameter-list> := <parameter>;[ <parameter-list> ]
<parameter> := <name>=<value>
```

where `<proxy-info>` is `<host>:<port> [ :<user>:<password> ]`, `<properties-file>` is the file imported during the setup, `<method>` is any HTTP command, such as GET, POST or DELETE, and `<params>` is a colon-separated list of parameter name and value which is expressed as `<name>=<value>`.

Invoking the restutil Configuration Tool

1. After installing core software (see "Installing Core Software"), create the configuration file and copy it to the `$ARCSIGHT_HOME` directory.

   The connector expects the properties file to be in the `$ARCSIGHT_HOME/user/agent/flexagent` directory.

2. From the `$ARCSIGHT_HOME/bin` directory, run the `arcsight restutil` script.

Retrieving an Access Token

To retrieve an access token, invoke the `arcsight restutil` script with the `token` command:

```
arcsight restutil token [ -proxy <proxy-info> ] -config <properties-file>
```

where `<proxy-info>` is `<host>:<port> [:<user>:<password>]` and `<properties-file>` is the file imported during configuration.

For example, the following command will print the token to the console:

```
# arcsight restutil token -proxy proxy.abc.com:8080\ -config
google.properties
. . .
```

```
ya29.AHES6ZQp0jYZrsQYdGz2Nk0HAGfI3_AjTtxwOpeXywtZ-E_gpjQIDw
```

⬅ access token

Retrieving Values from the Server

To retrieve values from the server, invoke the `arcsight restutil` script with the `execute` command. A valid access token should be specified if the response is to be sent only to the authenticated user.

```
arcsight restutil execute [ -proxy <proxy-info> ] -url "<url>" [ -token
<token> | -method <method> | -header "<params>" | -query "<params>" | -
content "<params>" ]
```

This command can be used to run authorized and unauthorzied commands. To run a command in an authorized way, obtain an access token with the `arcsight restutil token` command and pass the token using the `-token` option. In this case, the body of HTTP response will be displayed in the console. For example:

```
# arcsight restutil execute -token <token>\
-proxy proxy.abc.com:8080\
-url "https://apps-apis.google.com/a/feeds/customer/2.0/customerId?alt=json"
. . .
{"version":"1.0","encoding":"UTF-8","entry":
{"xmlns":"http://www.w3.org/2005/Atom","xmlns$apps":"http://schemas.google.co
m/apps/2006",
"id":{"$t":"https://apps-
apis.google.com/a/feeds/customer/2.0/C03sabdbl"},"updated":{"$t":"2013-02-
22T20:23:34.364Z"},
"link":[{"rel":"self","type":"application/atom+xml","href":"https://apps-
apis.google.com/a/feeds/customer/2.0/C03sabdbl"},
{"rel":"edit","type":"application/atom+xml","href":"https://apps-
apis.google.com/a/feeds/customer/2.0/C03sabdbl"}],
"apps$property":
[{"name":"customerOrgUnitDescription","value":"archp.mygbiz.com"},
{"name":"customerId","value":"C03sabdbl"},
{"name":"customerOrgUnitName","value":"archp.mygbiz.com"},
{"name":"description","value":""},"name":"name","value":"archp.mygbiz.co
m"}]}}
```

Retrieving Values Using an Authorized GET Command

Invoke the `arcsight restutil` script with the `authget` command to retrieve the result through an authorized GET command. This combines two commands. First, it obtains the access token and then executes the authorized GET command. The HTTP response will be displayed in the console. For example:

```
arcsight restutil authget [ -proxy <proxy-info> ] -config <properties-file> -
url "<url>"
```

For example, the event URL for Google includes the customer id in the path. The following command returns the JSON formatted response that has the customer information. `customerID` should be part of the response.

```
# arcsight restutil authget -proxy proxy.abc.com:8080\
-config google.properties
-url "https://apps-apis.google.com/a/feeds/customer/2.0/customerId?alt=json"
. . .
{"version":"1.0","encoding":"UTF-8","entry":
{"xmlns":"http://www.w3.org/2005/Atom","xmlns$apps":"http://schemas.google.co
m/apps/2006",
"id":{"$t":"https://apps-
apis.google.com/a/feeds/customer/2.0/C03sabdbl"},"updated":{"$t":"2013-02-
22T20:13:22.646Z"},
"link":[{"rel":"self","type":"application/atom+xml","href":"https://apps-
apis.google.com/a/feeds/customer/2.0/C03sabdbl"},
{"rel":"edit","type":"application/atom+xml","href":"https://apps-
apis.google.com/a/feeds/customer/2.0/C03sabdbl"}],
"apps$property":
[{"name":"customerOrgUnitDescription","value":"archp.mygbiz.com"},
{"name":"customerId","value":"C03sabdbl"},
{"name":"customerOrgUnitName","value":"archp.mygbiz.com"},
{"name":"description","value":""},
{"name":"name","value":"archp.mygbiz.com"}]}}
```

Send Documentation Feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this computer, click the link above and an email window opens with the following information in the subject line:

Feedback on Developer's Guide (FlexConnector for REST)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to Documentation-Feedback@microfocus.com.

We appreciate your feedback!