OAuth - Google - Service Account

Introduction

The Jitterbit OAuth - Google - Service Account plugin is used to authenticate with Google's implementation of OAuth 2 for Google service accounts. A step-by-step guide for configuring a Google service account is available in Google's documentation Using OAuth 2.0 for Server to Server Applications.

After the plugin is executed in a Jitterbit Script, the authentication token is returned in the global variable `google.oauth.token.auth` and the expiration number of seconds is returned in the global variable `google.oauth.token.expireSeconds`.

These global variables can then be used in other parts of the project. For example, `google.oauth.token.auth` can be used within an HTTP URL or header to provide the authentication details to Google. The variable `google.oauth.token.expireSeconds` can be used to cache the authorization token until the number of expiration seconds is met.

Usage

While this plugin can be executed in any Jitterbit Script, the global variables it returns are supported within the configuration of these endpoints:

- Design Studio HTTP Source
- Design Studio HTTP Target
- Cloud Studio HTTP Connection
- Cloud Studio HTTP Activity

The plugin output is not supported in Cloud Studio Connector Builder. Instead, in Cloud Studio you can use the Google Docs, Google Drive, or Google Sheets connectors, which already provide a means of authentication with the Google service during the configuration of the endpoint.

Steps

This page details the steps to use the OAuth - Google - Service Account plugin:

1. Associate the Plugin with an Agent Group
2. Set Global Variables for the Plugin
3. Execute the Plugin in a Jitterbit Script
4. Complete the Operation Setup

Example

A complete example project is provided in the Design Studio Jitterpak jb.radpak.google (JPK). This Jitterpak is self-documented by comments within each Jitterbit Script contained within the project.

On This Page

Plugin

- OAuth - Google - Service Account 1.0.0.0 (ZIP)

Jitterpak

- jb.radpak.google (JPK)

Related Articles

- Customizations > Plug-ins
- Plugins Available in Jitterbit Harmony

Related Topics

- Apply Plug-ins (Design Studio)
- HTTP (Cloud Studio)
- HTTP (Design Studio)
- Plugin Library
- Plugins (Cloud Studio)

Last updated: Jan 31, 2020

Associate the Plugin with an Agent Group

If using a Cloud Agent Group, you can skip this step, as the Jitterbit OAuth - Google - Service Account plugin is already associated with Jitterbit's Cloud Agent Groups by default.

If using a Private Agent Group, you must associate this Jitterbit-provided plugin with your Private Agent Group. This will make the plugin available on all Private Agents within the Private Agent Group. To associate the plugin:

1. Log in to the Harmony Portal and go to the **Management Console**. Then use the menu in the top left to navigate to **Customizations** > **Plug-ins**.
2. In the table, locate the row "Jitterbit OAuth - Google - Service Account." On the far right, use the **Action** dropdown to select **Associate Agent Group**.
3. In the popup, select your Private Agent Group and click **Save**.

TIP: Detailed documentation on associating plugins is available in Customizations > Plug-ins.

Set Global Variables for the Plugin

In order for this plugin to be functional, global variables required as input by the plugin must be set.

These global variables can be set in the same script that executes the plugin (covered next in Execute the Plugin in a Jitterbit Script), or they can be set in a separate script that is located earlier in the operation chain (so that the variables are initialized prior to the execution of the plugin).

TIP: For more information, see these pages:

- Creating a Script (Design Studio)
- Global Variables (Design Studio)
- Script Types and Creation (Cloud Studio)
- Global Variables (Cloud Studio)

A script template is provided below, followed by documentation on each of the input global variables that can be used with the plugin:

Setting Input Variables

```
<trans>
// Input variables
$google.oauth.privateKey = "-----BEGIN PRIVATE KEY-----\nABCDEfg...
HIJKLMN\n-----END PRIVATE KEY-----\n";
$google.oauth.clientEmail = "example@example-123456.iam.gserviceaccount.
com";
$google.oauth.accountScopes = "https://www.googleapis.com/auth/bigquery";
$google.oauth.serviceAccountUser = '';
</trans>
```

Name	Type	Required	Description
`google.oauth.privateKey`	String	**Required**	The RSA private key that is given via JSON in the "private_key" field from Google.
`google.oauth.clientEmail`	String	**Required**	The client email given via JSON in the "client_email" field from Google.
`google.oauth.accountScopes`	String	Optional	The space-separated OAuth scopes to use with the service account flow. Set to an empty string for none.
`google.oauth.serviceAccountUser`	String	Optional	The email address of the user the application is trying to impersonate in the service account flow. Set to an empty string for none.

TIP: For additional details, refer to Google's documentation Using OAuth 2.0 for Server to Server Applications and OAuth 2.0 Scopes for Google APIs.

Execute the Plugin in a Jitterbit Script

This plugin can be executed from a Jitterbit Script of any script type by calling the Jitterbit Script function `RunPlugin()`.

The global variables needed as input for the plugin can be specified within the same script that the plugin is executed within. For example, the contents of the script used for the template in Set Global Variables for the Plugin could be included in a single script that also runs the plugin.

After the plugin is executed, the authentication token is returned in the global variable `google.oauth.token.auth` and the expiration number of seconds is returned in the global variable `google.oauth.token.expireSeconds`.

Each of these parts of the script are provided in this template:

Setting Input Variables and Executing the Plugin

```
<trans>
// Input variables
$google.oauth.privateKey = "-----BEGIN PRIVATE KEY-----\nABCDEfg...
HIJKLMN\n-----END PRIVATE KEY-----\n";
$google.oauth.clientEmail = "example@example-123456.iam.gserviceaccount.
com";
$google.oauth.accountScopes = "https://www.googleapis.com/auth/bigquery";
$google.oauth.serviceAccountUser = '';

// Output variables
$google.oauth.token.auth = '';
$google.oauth.token.expireSeconds = '';

// Executing the plugin
RunPlugin("<TAG>plugin:http://www.jitterbit.com/plugins/pipeline/user
/OAuthGoogleServiceAccount</TAG>");
</trans>
```

Name	Description
`google.oauth.`**`token.auth`**	The Google authentication token.
`google.oauth.`**`expireSeconds`**	The number of seconds until the Google authentication token expires.

TIP: Detailed documentation on executing plugins from a script is is available in Apply Plug-ins (Design Studio) or Plugins Called in a Script (Cloud Studio).

Complete the Operation Setup

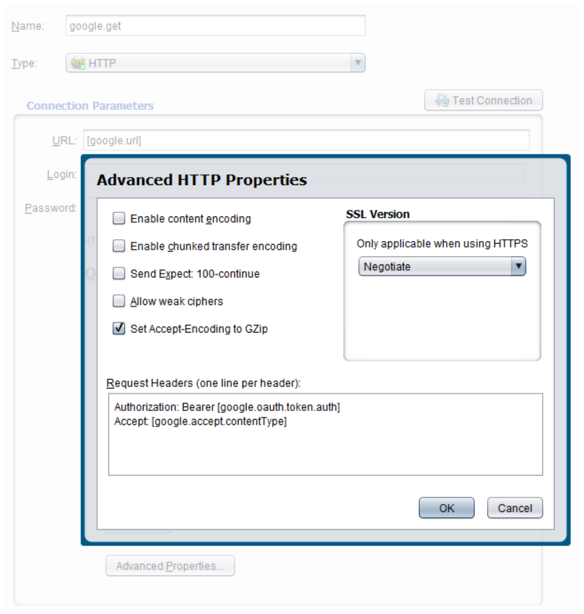
The global variables returned by the plugin script can then be used in other parts of the project for authentication with Google. These global variables are officially supported within the configuration of these endpoints:

- Design Studio HTTP Source
- Design Studio HTTP Target
- Cloud Studio HTTP Connection
- Cloud Studio HTTP Activity

In a typical configuration in Design Studio, the global variable `google.oauth.token.auth` can be used within an HTTP URL or header to provide the authentication details to Google.

This can be concatenated with the Google host in the **URL** field of an HTTP source or HTTP target as shown in the example Jitterpak jb.radpak.google (JPK).

Alternatively, it could be entered in the HTTP source or target **Advanced HTTP Properties** in the **Request Headers** area:



Name: google.get

Type: HTTP

URL: [google.url]

Login:

Password:

- [ ] Enable content encoding
- [ ] Enable chunked transfer encoding
- [ ] Send Expect: 100-continue
- [ ] Allow weak ciphers
- [x] Set Accept-Encoding to GZip

SSL Version

Only applicable when using HTTPS

Negotiate

Request Headers (one line per header):

```
Authorization: Bearer [google.oauth.token.auth]
Accept: [google.accept.contentType]
```

Similarly, in Cloud Studio, the global variable `google.oauth.token.auth` can be used in an HTTP connection within the **Request Headers** field. Alternatively it can be specified as either a **Request Parameter** or a **Request Header** in an HTTP activity.

A complete example project using this plugin is provided in the Design Studio Jitterpak jb.radpak.google (JPK). This Jitterpak also includes an example of using the global variable `google.oauth.token.expireSeconds` to cache the authorization token until the number of expiration seconds is met. The Jitterpak is self-documented by comments within each Jitterbit Script contained within the project.