December, 2014

Installation Jython Scripts

This document provides reference information and examples relating to the installation jython scripts used in the Community Manager installation procedures.

It includes:

Jython Script Elements

The various jython script elements are the key to a successful installation. Some of the elements include built-in flexibility that enables you to configure your installation in different ways. It's a good idea to review the available options before installing.

Information on the various script elements is given below, with an explanation of each.

--url

The base url of the Platform API. The URL is normally structured along these lines:

http://[hostname]:9900

It is the hostname that the SOA container is running on. There is normally no context unless the product is running in an application server.

Example: http://enterprise.soa.local:9900

--tenantName

A friendly name for the tenant, for use in certain scenarios such as email messages.

--tenantId

The internal ID of the tenant. It cannot have spaces or special characters, and should be lowercase. The tenantID is normally the lowercase version of the tenant name above (without spaces). It will appear in all object IDs and the URLs in the system.

--address

The base URL of the tenant. The hostname must be unique. This hostname what will be used in the browser when accessing the UI, and the product uses it to identify the tenant. There is normally no context unless the product is running in an application server.

Example: http://enterpriseapi.soa.local:9900

--consoleAddress

The same as the --address element, but includes the context in which Community Manager is running. This is the full URL that will be used in the browser when accessing the UI.

Example: http://enterpriseapi.soa.local:9900/enterpriseapi

--theme

The UI theme identifier. Unless a custom theme has been developed, the value is **default**.

See also related optional element *themeImpl* below.

For information about how the **theme** and **themeimpl** elements are used together to provide customizability, see Cloning Themes: themeImpl Install Parameter on page 2.

--email

The email address you want to use as the default tenant administrator address.

Note: If you don't include this, the following default value is assigned: administrator@{tenantid}.

--password

The password you want to configure for the default tenant administrator.

--contactEmailAddress

Used in email templates.

--fromEmailAddress

Account used by the system to send email.

--virtualHosts

A comma-separated list of host names that the product will accept. For example:

--virtualHosts open.acmepaymentscorp.com,open.example.com

--themeImpl

An optional element that allows the configuration theme name (value for the **--theme** element above) and the theme implementation name to be different. Two possible values:

--theme mytheme --themeImpl default

--theme mytheme --themeImpl simpledev

For information about how the **theme** and themeimpl elements are used together to provide customizability, see Cloning Themes: themeImpl Install Parameter on page 2.

Cloning Themes: themeImpl Install Parameter

In versions of the platform before 7.2, a single tenant could only have one theme (codebase) for a single implementation, and one set of customizations that could be applied to that theme.

In most cases, customers used the **default** theme and customized it via the **custom.less** file. In rare cases, the customer had a second theme as well—but each theme could only have one set of customizations applied to it, via the **custom.less** file.

Version 7.2 introduces a new installation parameter, **themeImpl**, to the jython script used to configure the tenant. Using the new parameter in conjunction with the **theme** parameter, it is possible to clone a specific theme implementation with two or more user-defined sets of customizations. One set of customizations is independent of another.

The new parameter takes the definitions previously applicable to the **theme** parameter. Currently, there are two valid values for **themeImpl**:

- **default** (used for the main Developer Portal)
- **simpledev** (used for the new Simple Developer theme)

The **theme** parameter now takes a customer-defined value, naming a theme that the customer can then customize via the **custom.less** file. The customer can define multiple values for the **theme** parameter; each value must be unique for the tenant.

For backwards compatibility, if the **themeImpl** parameter is not present in the installation jython script, the value provided for the **theme** parameter in the installation script must be one of the valid core values of **default** or **simpledev**.

By using these two parameters together, customers can define multiple access points into the platform, each of which can have its own set of customizations applied. Some sample scenarios:

- **Multiple customizations of look and feel (not previously supported)**: Tenant ACMEPayments Corp has 10 partners; each partner has a different API on the platform, and each has an API admin who accesses the platform and exports metric information for the API.

  When installing version 7.2, the site admin defines the **themeImpl** parameter as **default** and creates 10 themes, each named for one of the 10 partners, and each with a different URL that includes the tenant name and partner name.

  After installation, the site admin customizes each version by uploading a **custom.less** file for each partner with the partner's company colors and logo.

  When each API admin logs in via the URL for his company, he sees the colors and logo for his company.

- **Two theme implementations, additional theme customizations**: Tenant ACMEPaymentDev Inc. has two audiences, a developer audience and an API Admin audience.

  In previous versions, this scenario was supported in a customer case by supporting two completely different theme implementations (one of which was a completely customer-specific customization). Although this solution allowed support of two themes, it was not extensible.

  Now, the same solution can be accomplished for any customer by supporting two themes, default and simpledev. But each theme can also be cloned, with additional values provided for the new themeImpl parameter, allowing additional customization to be supported.

Two-theme example: when installing version 7.2, the site admin defines the **themeImpl** parameter as **default** and then again as **simpledev**, and creates two themes:

- **developer** theme uses the **simpledev** theme implementation
- **apiadmin** theme uses the **default** theme implementation

He does not bother with a **custom.less** file since having a different look and feel is not significant at the moment, but this can easily be added later at any point.

Developers see the simpler user interface presented by the **simpledev** theme implementation, while API Admins have access to the API management functionality offered only in the **default** theme implementation.

Using theme and themeimpl in the Installation Script

The points below summarize the script changes.

- New parameter **themeImpl** takes one of the following values previously assigned to the **theme** parameter: **default** or **simpledev**. The **theme** parameter is now user-defined; can take any name, as defined by the tenant admin. The theme can be customized later.
- Backwards compatibility: if the **themeImpl** parameter is not passed, the **theme** parameter must be set to one of the following values: **default** or **simpledev** (same as older functionality).

First, create a new tenant with a single theme, using the standard jython script. An example is shown below.

```
jython ../scripts/Lib/soa/atmosphere/tenant.py -a -v --url http://localhost:<port> --tenantId <tenantId> --tenantName <tenantName>  --address http:// < hostname >:<port> --theme <anyname> --themeImpl <themename>  --consoleAddress http://<hostname>:<port>/atmosphere/
```

Then, for each tenant, run a database script (see next section), choosing a value for **theme** and **themeImpl**. In a scenario where you are cloning themes for the same tenant, you will need to modify the script appropriately and run it several times, using the same value for **themeImpl** and a unique value for **theme**, as well as a unique URL, in each script.

Each instance with a separate **theme** value can be customized later with a **custom.less** file.

How It Works

To create a new tenant with a custom theme name, there are two steps. First, create the tenant with a single theme using the jython script. Then, either clone an existing theme or add a new theme by using the **theme** and **themeImpl** parameters together, using an additional database script.

To create a new tenant with a custom theme name

1 Create a tenant with a single theme using the jython script. An example is shown below.

```
jython ../scripts/Lib/soa/atmosphere/tenant.py -a -v --url http://localhost:<port> --tenantId <tenantId> --tenantName <tenantName>  --address http:// < hostname >:<port> --theme <anyname> --themeImpl <default/simpledev>  --consoleAddress http://<hostname>:<port>/atmosphere/
```

2 Run the database script, as in the example below, to add themes or clone themes using the **theme** and **themeImpl** parameters together.

Example: Customizing the Script

The table below shows a test scenario where four separate URLs, with different themes, are provided for the same tenant.

To implement this scenario, first run the standard jython script.

Tenant	Theme	Themeimpl	URL	Comments
open	default	default	http://acmepaymentscorpdefault/atmosphere	New tenant created by using new script and by passing themeImpl and theme parameters.
open	opendeveloper	Default	http://acmepaymentscorpdev/atmosphere	Changes done to TENANT and TENANT_THEME tables using SQL script.
open	openadmin	default	http://acmepaymentscorpadmin/atmosphere	Changes done to TENANT and TENANT_THEME tables using SQL script.
open	opensimpledev	Simpledev	http://acmepaymentscorpsimpledev/atmosphere	Changes done to TENANT and TENANT_THEME tables using SQL script.

Once the tenant has been created, running the customized sample script shown below implements the scenario shown in the table above.

```
insert into tenant_themes (tenantid, theme,virtualhost,consoleaddress,themeImpl)
values((select tenantid from tenants  where
fedmemberid='open'),'openadmin','openadmin.soa.dev','http://acmepaymentscorpadmin/atmosphere','default');

insert into tenant_themes (tenantid, theme,virtualhost,consoleaddress,themeImpl)
values((select tenantid from tenants  where
fedmemberid='open'),'opendeveloper','opendeveloper.soa.dev','http://acmepaymentscorpdev/atmosphere','default');

insert into tenant_themes (tenantid, theme,virtualhost,consoleaddress,themeImpl)
values((select tenantid from tenants  where
fedmemberid='open'),'opensimpledev','opensimpledev.soa.dev','http://acmepaymentscorpsimpledev/atmosphere','sim
pledev');

update tenants set virtualhost = 'open.soa.dev,openadmin.soa.dev,opendeveloper.soa.dev,opensimpledev.soa.dev'
where fedmemberid='open'
```

Installation Scripts with Examples

From the /bin folder of your installation, run the applicable jython script below, using appropriate values for your installation. For definitions, refer to the table above.

Windows

Script:

```
Jython.bat ../scripts/Lib/soa/atmosphere/tenant.py -a -v --url [url] --tenantName [tenantName] --tenantId [tenantId] --
address [tenantAddress] --consoleAddress [consoleAddress] --theme [theme] --themeImpl [themeImpl] --email
[default tenant administrator] --password [default tenant administrator password] ----contactEmailAddress
[contactEmailAddress] --fromEmailAddress [fromEmailAddress] --virtualHosts [virtualHosts]
```

Example:

```
jython.bat ../scripts/Lib/soa/atmosphere/tenant.py -a -v --url http://enterprise.soa.local:9900 --tenantName
EnterpriseAPI--tenantId enterpriseapi --address http://enterpriseapi.soa.local:9900 --consoleAddress
http://enterpriseapi.soa.local:9900/enterpriseapi --theme default --themeImpl [default] --email
administrator@<yoursite>.com --password password --contactEmailAddress yourname@acmepaymentscorp.com --
fromEmailAddress yourname@acmepaymentscorp.com
```

UNIX

Script:

```
/jython.sh../scripts/Lib/soa/atmosphere/tenant.py -a -v --url [url] --tenantName [tenantName] --tenantId [tenantId] --
address [tenantAddress] --consoleAddress [consoleAddress] --theme [theme] --themeImpl [themeImpl] --email
[default tenant administrator] --password [default tenant administrator password] ----contactEmailAddress
[contactEmailAddress] --fromEmailAddress [fromEmailAddress] --virtualHosts [virtualHosts]
```

Example:

```
./jython.sh ../scripts/Lib/soa/atmosphere/tenant.py -a -v --url http:// enterpriseapi.soa.local:9900 --tenantName
EnterpriseAPI --tenantId enterpriseapi--address http://enterpriseapi.soa.local:9900 --consoleAddress
http://enterpriseapi.soa.local:9900/enterpriseapi --theme default --themeImpl default --email
administrator@<yoursite>.com --password password --contactEmailAddress yourname@acmepaymentscorp.com --
fromEmailAddress yourname@acmepaymentscorp.com
```

Secure Installation: Steps before Running Jython Scripts

If you will be using HTTPS URLs for your Community Manager installation, there are some additional steps you must perform **before** running the jython script. If these setup steps are not in place, the jython script will fail.

The jython installation script invokes a REST API call on Community Manager. If the Community Manager container is configured with HTTPS, the jython script must use a keystore with a trusted certificate; otherwise, it will not be able to connect to Community Manager at all.

If the certificate is a default trusted certificate issued by one of the trusted CAs already recognized by the JRE, no extra steps are needed. However, if the Community Manager container is configured with HTTPS, you'll need to add the issuer to the certificate file before running the jython script if your Community Manager SSL certificate meets any of these conditions:

- The certificate is self-signed.
- The certificate is generated by the SOA platform's internal CA.
- The certificate is generated by a non-commercial issuer.

Setting up security for a Community Manager tenant with HTTPS URLs

To add the issuer to the certificate file you must get the certificate issuer of the container, set the path, and then set the jython ops command. You can then run the jython script. Follow the steps below.

To set up security for a CM tenant with HTTPS URLs

1 In SOA Software Policy Manager, create a keystore with the issuer certificate of Community Manager SSL Certificate.

You could use a tool such as the Java Keytool utility (http://sourceforge.net/projects/keytool) or the Elysian Keytool Advanced GUI (http://sourceforge.net/projects/ssltools) to create the keystore. An example using Keytool is shown below:

```
keytool -genkey -alias mydomain -keyalg RSA -keystore keystore.jks -keysize 2048
```

2 Import the trusted certificate into the keystore. An example is shown below. Substitute values such as filename and location and keystore name and location.

```
keytool -importcert -v -trustcacerts -alias atmo -file C:/downloads/CA.cer -keystore c:/downloads/keystore.jks -storepass password -storetype JKS
```

Once this step is complete, the keystore is ready with the Community Manager SSL certificate.

Note: As an alternative to the above step, which uses the default cacerts keystore and adds the new certificate issuer, you could create a blank cacerts file with just this issuer, the Community Manager certificate issuer. If you take this approach you would still have to follow Step 3 below, and then run the jython script.

3 In the **bin** folder for your installation, at the command line, execute the following command, substituting the keystore path and filename:

```
set JYTHON_OPTS="-Djavax.net.ssl.trustStore={keystore_path_and_filename}"
```

An example is shown below.

```
set JYTHON_OPTS="-Djavax.net.ssl.trustStore=C:/testdata/mykeystore.jks"
```

4 Run the jython script with the values for your installation.

Windows:

```
jython.bat ../scripts/Lib/soa/atmosphere/tenant.py -a -v --url https://enterprise.soa.local:9443 --tenantName EnterpriseAPI--tenantId enterpriseapi --address https://enterpriseapi.soa.local:9443 --consoleAddress https://enterpriseapi.soa.local:9443/enterpriseapi --theme default --email administrator@<yoursite>.com --password password --contactEmailAddress yourname@acmepaymentscorp.com --fromEmailAddress yourname@acmepaymentscorp.com
```

UNIX:

```
./jython.sh ../scripts/Lib/soa/atmosphere/tenant.py -a -v --url https:// enterpriseapi.soa.local:9443 --tenantName EnterpriseAPI --tenantId enterpriseapi--address https://enterpriseapi.soa.local:9443 --consoleAddress https://enterpriseapi.soa.local:9443/enterpriseapi --theme default --email administrator@<yoursite>.com --password password --contactEmailAddress yourname@acmepaymentscorp.com --fromEmailAddress yourname@acmepaymentscorp.com
```

An example is shown below.

```
jython ../scripts/Lib/soa/atmosphere/tenant.py -a -v --url https://acmepaymentscorp.com:9443 --tenantName
admin@acmepaymentscorp --tenantId admin@acmepaymentscorp --address
https://acmepaymentscorp.com:9443 --consoleAddress https://acmepaymentscorp.com:9443/atmosphere --
theme default --contactEmailAddress admin@acmepaymentscorp.com --fromEmailAddress
admin@acmepaymentscorp.com --virtualHosts acmepaymentscorp.com
```

If you are not sure how to do SSL configuration for your listener, follow the instructions below.

Configure HTTPS listener when using the SOA Software Platform internal Certificate Authority

To configure the HTTPS listener, there are three basic steps:

1 In the SOA Software Policy Manager Console, configure the certificate authority.

See Step 1: In Policy Manager, configure the Certificate Authority on page 8.

2 Add the HTTPS listener to CM container.

See Step 2: Add HTTPS listener to CM container on page 8.

3 Manage PKI keys for the HTTPS Listener.

See Step 3: Manage PKI Keys for HTTPS Listener on page 9.

Step 1: In Policy Manager, configure the Certificate Authority

The first step is to configure the Certificate Authority.

To configure the CA in Policy Manager

1 In Policy Manager, go to Configure > Security > Certificates > Certificate Authority.
2 Click **Configure Certificate Authority**.
3 At the **Select Certificate Configuration Option** page, click **Generate X.509 CA Certificate and PKI Keys**.
4 At the **Generate CA Certificate and PKI Keys** page, provide values and then click **Finish**.
5 At the summary page, click **Export CA Certificate**. Save it to a location on your local filesystem.

Step 2: Add HTTPS listener to CM container

Next, you must add an HTTPS listener to the container.

To add an HTTPS listener to the CM container

1 In the SOA Software Policy Manager Workbench, go to SOA Software Community Manager > Containers > CM Container > Details > Inbound Listeners.
2 Click **Add Container Listener**.

3 At the **Select Container Listener Type** page, select HTTPS and click **Next**.

4 Provide values for the listener: Name and Description, Bind to all interfaces checkbox, Host Name, Port Number, Content Path, and thread pool details.

5 Click **Finish**. The HTTPS listener is added.

Step 3: Manage PKI Keys for HTTPS Listener

Your subject DN must match with the CA, and the HTTPS listener certificate must use the same hostname that you are using, otherwise the jython script will fail.

Follow the steps below.

To manage PKI keys for the HTTPS Listener

1 In the Policy Manager Organization Tree, under Community Manager > Containers > CM Container > Details > Inbound Listeners, from the Actions drop-down menu, click **Manage PKI Keys**.

2 In the **Select Key Management Options** page, under Key Management Options, choose **Generate PKI Keys and X.509 Certificate**. Click **Next**.

3 In the **Generate PKI Keys & X.509 Certificate** page, provide the same details you set up for the Certificate Authority's Subject DN (see To configure the CA in Policy Manager, Step 3).

4 Click **Finish**.

5 In the summary page, in the Certificate Details section, verify that the values for Subject DN and Issuer DN match.

–♦–