CentraSite

Registry Federations in CentraSite

Version 9.5 SP1

November 2013

This document applies to CentraSite Version 9.5 SP1.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: IINM-AG-FEDREP-95SP1-20140410

Table of Contents

Preface

CentraSite provides support for federated registries. Federation allows you to obtain copies of objects from one registry and store them in another registry, where you intend to access and navigate them.

The registries participating in a federation can be CentraSite registries and UDDI v3-compliant registries. Currently, the UDDI federation support allows federating data from a 3rd party UDDI registry such as jUDDI or SAP to CentraSite.

For general information about UDDI v3, refer to the specification at **http://uddi.org/pubs/uddi_v3.htm**.

The content is organized under the following sections:

Overview of the Federation Process	Gives an overview of the federation process and describes the steps involved in setting up and running a federation.
Creating a Federation Configuration File	Describes how to create the configuration file containing the technical setup parameters for the federation.
Adding a UDDI Federation to the Federation List	Describes how to register a new federation configuration.
Triggering the UDDI Federation	Describes how to activate a federation.
Removing a UDDI Federation from the Federation List	Describes how to unregister a federation configuration.
Displaying the List of Target Registries defined for Federation	Describes how to determine which target registries are defined for federation.
Mapping UDDI Object IDs and tModels	Describes how to maintain uniqueness of UDDI object IDs during the federation process, and how to map UDDI structures onto JAXR-based structures in the target registry.
Users and Access Rights	Summarizes the user permissions required in order to perform a federation.

1 Overview of the Federation Process

The UDDI-based federation allows you to federate data between two UDDI v3 compliant registries. The data to be federated is inquired from the source UDDI registry using the UDDI Inquiry API. The obtained data is transformed into one or more XML-formatted events whose structure corresponds to an event type in the Event Type Store, and the events are emitted across the event bus. When an event reaches the UDDI endpoint, the data to be federated is extracted from the event and published to the target registry using the UDDI publish API.

The federation process consists of the following steps:

- Creating a federation configuration file that specifies the source and target registries and the objects you want to copy. This file is used as input for the commands to add and trigger the federation.
- Adding the federation configuration to the list of target registries defined for federation.
- Triggering the federation as required. This can be either on an ad hoc basis, or using the scheduling mechanism of your operating system.
- When a federation is no longer required, it can be removed from the list of target registries defined for federation.

You can carry out these steps using the command line script `CentraSiteCommand`, which is located in *<CentraSiteInstallDir>/utilities*. See the following sections for details of usage.

Federation is carried out from a source UDDI registry to one or more target CentraSite registries. Typically, the source and target registries are on different machines, and they are addressed by their URL. The federation commands (create, add, trigger, etc.) must be run on a machine where CentraSite is installed, i.e. on one of the target machines.

The list of target registries defined for federation is maintained in a file named *emit.xml* on the target machine where the federation commands are run. This file is located at *<CentraSiteInstallDir>/Federation/config/templates/emit.UddiFederation/OSGI-INF/blueprint/*.

When you add a federation, details of the federation are added to *emit.xml*, and when you trigger a federation, the information contained in *emit.xml* is used to route the federated data to the target registry. Thus, the commands for adding and triggering a federation must run on the same machine.

If there is more than one target registry, each machine that hosts a registry can in theory contain its own *emit.xml* file with its own list of target registries defined for federation. However, it is generally simpler to maintain an *emit.xml* file on just one of the target machines, and to trigger all of the federations as required from that machine.

Note that there is no automatic synchronization between the *emit.xml* files across multiple target registries; entries made in one *emit.xml* file are not copied automatically to other *emit.xml* files.

2 Creating a Federation Configuration File

To configure the UDDI federation, you need to supply the following details:

- The source registry details where the original data is located: host, port, username, password (needed for connecting to the registry for inquiring data).

  Also the URLs of the UDDI security and inquiry APIs, to allow the UDDI Client API to connect to the registry.
- The target registry details to which the data will be copied: host, port, username, password (needed for connecting to the registry for publishing data).

  Also the URLs of the UDDI security, inquiry and publish APIs, to allow the UDDI Client API to connect to the registry.
- The data that has to be inquired from the source registry. The types of UDDI object that can be federated are business entities, business services and tModels. Refer to the standard UDDI specification for details of these object types.
- The location of the Event Type Store. This is by default `<SAGInstallDir>`/*common/EventTypeStore/*.
- The location where further internal components required for the federation are available. This is by default `<SAGInstallDir>`/*CentraSite/Federation*.

You can configure these details in an XML configuration file. A template of the XML configuration file is available in `<SAGInstallDir>`/*CentraSite/Federation/config/federationConfiguration.xml*. You can copy this file and edit it to suit your needs.

After configuring these details, you can start the federation.

Here is a sample configuration file:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:federationConfiguration
xmlns:ns2="http://namespaces.softwareag.com/EDA/Federation/UDDI/Configuration"
xmlns:ns3="urn:uddi-org:api_v3">
 <sourceRegistryDetail>
  <host>MyUDDIServer</host>
  <port>8080</port>
  <user>root</user>
  <password>root</password>
  <securityUrl>http://MyUDDIServer:8080/juddiv3/services/security</securityUrl>
  <inquiryUrl>http://MyUDDIServer:8080/juddiv3/services/inquiry</inquiryUrl>
 </sourceRegistryDetail>

 <targetRegistryDetail>
  <host>FedTarget</host>
  <port>53307</port>
  <user>AdminUser</user>
  <password>ABCXYZ123</password>
  <defaultBusinessEntity>Default Organization</defaultBusinessEntity>
  <securityUrl>http://FedTarget:53307/UddiRegistry/security</securityUrl>
  <inquiryUrl>http://FedTarget:53307/UddiRegistry/inquiry</inquiryUrl>
  <publishUrl>http://FedTarget:53307/UddiRegistry/publish</publishUrl>
 </targetRegistryDetail>

 <inquiryParams>
  <find_business>
   <ns3:findQualifiers>
    <ns3:findQualifier>caseInsensitiveMatch</ns3:findQualifier>
    <ns3:findQualifier>approximateMatch</ns3:findQualifier>
   </ns3:findQualifiers>
   <ns3:name>Custom Organization</ns3:name>
  </find_business>

  <find_service>
   <ns3:findQualifiers>
    <ns3:findQualifier>caseInsensitiveMatch</ns3:findQualifier>
    <ns3:findQualifier>approximateMatch</ns3:findQualifier>
   </ns3:findQualifiers>
   <ns3:name>PlainService</ns3:name>
   <ns3:name>TestService</ns3:name>
  </find_service>

  <find_tModel>
   <ns3:findQualifiers>
    <ns3:findQualifier>caseInsensitiveMatch</ns3:findQualifier>
    <ns3:findQualifier>approximateMatch</ns3:findQualifier>
   </ns3:findQualifiers>
   <ns3:name>%</ns3:name>
  </find_tModel>
 </inquiryParams>

 <eventTypeStoreLoc>C:/SoftwareAG/common/EventTypeStore</eventTypeStoreLoc>
```

```
<nervAssetsLoc>C:/SoftwareAG/CentraSite/Federation/assets/uddi</nervAssetsLoc>

<sourceBasicAuthentication>true</sourceBasicAuthentication>

<targetBasicAuthentication>false</targetBasicAuthentication>

<batchEventEmission>true</batchEventEmission>

<batchPublish>false</batchPublish>

<timestampBasedUpdate>true</timestampBasedUpdate>

<eventEmissionTimeout>60</eventEmissionTimeout>

</ns2:federationConfiguration>
```

The source registry details are provided in the `sourceRegistryDetail` element.

The target registry details are provided in the `targetRegistryDetail` element. When the CentraSite registry is the target registry, the port number is by default 53307.

The `securityUrl` and `inquiryUrl` elements provide the location of the Security and Inquiry APIs that are required for accessing the UDDI registries. The `publishUrl` element for the target registry provides the location of the Publish API that is required for publishing the UDDI objects.

The `securityUrl`, `inquiryUrl` and `publishUrl` are optional for CentraSite target registries since these are the default settings.

If the configuration file does not contain the user and password details for the source and target registries, you will be prompted to supply these values when you initialize the federation.

The target registry has another configuration:

```
<defaultBusinessEntity>
Default Organization
</defaultBusinessEntity>
```

This option is to configure the default business Entity (Organization) in the target registry. All business services will be associated with a particular business entity in the source registry as shown here:

```
<businessService
  serviceKey="uddi:juddi.apache.org:68f6379d-9063-4eaf-b35a-45199c9f0b0a"
 businessKey="uddi:juddi.apache.org:4375a871-b13e-46ae-b329-c79871efc3f1">
```

The business key refers to a business entity in the source registry. When you try to publish this service to some other registry, it could be the case that the business entity to which this service is associated with may be already available in the target, in which case this service can be published as is in the target registry. If the target does not have the business entity to which the service is associated, you might still want to publish the service separately from the entity. The `defaultBusinessEntity` element provides the option to provide the default organization in the target registry to which the service must be associated when it is published. If the business key is present in the target, then the service is published without changes, and if the business key is not present, then the business key of the organization referred to in the XML file is associated with the service and it is published. In the example here, the "Default Organization" refers to the organization (business entity) in CentraSite.

Use the `inquiryParams` element in the configuration file to specify the objects that you want to select for federation.

Within this element, you can specify the following child elements:

- `find_business`: search for UDDI objects of type "business entity".
- `find_service`: search for UDDI objects of type "service".
- `find_tModel`: search for UDDI objects of type "tModel".

These are the UDDIv3 Inquiry API calls. Their syntax must adhere to descriptions in the section *Inquiry API Functions* of the UDDIv3 standard.

Not all of these child elements need to be supplied, so for example if you want to federate only business entities, it is sufficient to provide the `find_business` element.

The `name` element within the `find_business`, `find_service` or `find_tModel` element specifies the name of the object or objects to be searched for. You can use wildcards in the `name` element to search for multiple objects; these are the standard wildcards as described in the UDDI specification. For `find_business` and `find_service` you can supply more than one `name` element.

You can restrict the set of selected objects by using a filter, specified in a `findQualifier` element. The filter modifies the way in which the value in the `name` element is interpreted. The available filters are the findQualifiers as described in the UDDI specification. The example above illustrates how to use the findQualifiers to use case sensitivity/insensitivity, and exact/approximate match.

The location of the Event Type Store is given in the `eventTypeStoreLoc` element. Ensure that the location specified corresponds to your CentraSite installation path.

Internal components that are required for the federation are located in `nervAssetsLoc`. Ensure that the location specified corresponds to your CentraSite installation path.

Basic Authentication

The `sourceBasicAuthentication` and the `targetBasicAuthentication` elements define whether the authentication to the UDDI registry should be via authTokens or through HTTP Basic Authentication. They take Boolean values of either true or false.

Authentication to a UDDI registry usually works through the Security API of the UDDI standard through authTokens. The messages to UDDI are sent as SOAP messages over HTTP. Some UDDI registries provide an option to authenticate via HTTP Basic Authentication. CentraSite and jUDDI, for example, provide authentication via authTokens whereas SAP registry provides authentication via HTTP Basic authentication.

So the `sourceBasicAuthentication` element specifies this option for the source registry. If set to true, it means that the source will be authenticated via HTTP basic authentication and if set to false, the source will be authenticated via authTokens. The `targetBasicAuthentication` element configures the authentication method for the target registry. If set to true, authentication is done via HTTP Basic authentication and if set to false, the authentication is done via authTokens.

Note: Currently, the federation from an SAP source registry to a target CentraSite registry only supports HTTP Basic Authentication on the source SAP registry, i.e. authenticating via AuthToken on the source SAP registry is not supported. Therefore, the value in the `<sourceBasicAuthentication>` element must be "true". Similarly, if SAP is the target registry, then the value in the `<targetBasicAuthentication>` element must be "true".

Batch Event Emission

You can control whether the UDDI objects found by the inquiry calls should be sent individually to the target or as a batch containing all of the found objects. If you set `batchEventEmission` to true, the objects will be sent as a batch. If you set the value to false, the objects will be sent individually.

For example, if the inquiry calls return 10 UDDI objects for federation (these 10 objects could be businesses, services, tModels or mixture of all), and `batchEventEmission` is true, all 10 UDDI objects are emitted as one single XML event. If `batchEventEmission` is false, each of the 10 UDDI objects will be emitted individually.

Batch Publish

This configuration defines how the UDDI objects should be published. This is in alignment with the behavior of the UDDI Publish API.

If `batchPublish` is set to true, then events that were batched due to the setting of `batchEventEmission` will also be published on the target machine as a batch. If an error occurs while publishing one of the batch's objects, this object will not be published and all of the remaining objects in the batch will be ignored. This is the standard behavior of the UDDI Publish API.

If `batchPublish` is set to false, then events that were batched due to the setting of `batchEventEmission` will nevertheless be published individually on the target machine. If an error occurs while publishing one of the batch's objects, this object will not be published, but the processing of the remaining objects in the batch will continue.

Details of the publish operation will be provided in a log file.

Timestamp based update

This is an optimization configuration. During a federation operation, it can happen that an object being federated is already present in the target machine due to a previous federation operation. The following situations can occur:

- The object that is already available in the target registry might have been updated in the source registry and now this object has to be updated in the target registry as well.
- The object that is already available in the target registry has not been updated in the source registry, so the objects are identical.

Before an object is published is the target machine, CentraSite checks if the object already exists on the target machine. If the object already exists, but the timestamps are different, then the object is updated in the target registry. If the object to be published is identical to the object already present, the object is not updated. This results in performance optimization, thereby reducing the number of publish operations to the target registry.

If `timestampBasedUpdate` is set to true, then CentraSite performs the comparison and updates the target object if needed. If `timestampBasedUpdate` is set to false, the comparison is not performed and the object is published, regardless of whether the objects are updated objects or not.

Event Emission Timeout

The federation framework requires a certain amount of time for federating the data to the target registries and to publish the data in each of them. The time required for federation depends on the number of UDDI objects that have to be federated and the number of targets to which the data must be federated.

Example: If a single service is mentioned for federation, and this service has a large number of dependent tModels, then the time taken to federate these objects to a single target registry might be around 30 seconds, depending on various factors such system load on the source and target machines, and the speed of the network connection. If two target registries are involved, then the time taken would be doubled and so on. A service with a small number of dependent tModels might take just a few seconds to be published to a single target registry.

Occasionally a problem such as a broken network link will prevent the federation from working properly, and in this case the federation should terminate in an orderly fashion. For this reason, you can specify a timeout period for the federation. If the timeout period is exceeded, the federation will terminate. If the federation is started at regular intervals using the job scheduling mechanisms

of the operating system, the timeout only affects the currently started federation cycle; the operating system will start the next federation cycle as scheduled.

Make sure that the timeout value is sufficient to federate and publish the data to the target registries under normal processing conditions. If the timeout value is reached before all of the data has been federated, the remaining data will not be federated to the target registries.

Use the `eventEmissionTimeout` element to specify the timeout value in seconds. If this element is not specified, the default timeout value is 10 seconds.

3 Adding a UDDI Federation to the Federation List

After you have created the configuration file for a UDDI federation as described in the previous section, you can make the federation available for triggering by adding it to the list of target registries defined for federation. To do this, use a command of the following form:

```
CentraSiteCommand add Federation [-user <USER-ID>] [-password <PASSWORD>] -name
<NAME> -file <CONFIG-FILE>
```

The following table describes the complete set of input parameters that you can use with the `add Federation` utility:

Parameter	Description
`-user`	The user ID of a user on the target machine who has the permissions to create objects on the target machine.
`-password`	The password of the user identified by the parameter "-user".
`-name`	A name for the federation.
`-file`	The URI (file: or http:) of the configuration file.

For example:

```
CentraSiteCommand add Federation [-user "AdminUser"] [-password "ABCXYZ123"] -name
"FedName" -file "C:\Federation1.xml"
```

See the section **Users and Access Rights** below for more information. If the configuration file contains the user and password information in the `targetRegistryDetail`*`NAME`*, you can omit these from the command line. If you supply the user and password values on the command line and also in the configuration file, the values on the command line will be used.

Every federation has a dedicated name that you can provide in the *`Name`* parameter. This name is displayed when you use the `list Federation` command to show the list of target registries defined

for federation. It is also required when you wish to remove a federation using the `removeFederation` command.

When the federation has been added to the federation list, you can trigger the federation at any time, as described in the following section.

When you add a UDDI federation as described above, CentraSite creates an entry for the federation in the *emit.xml* file.

4 Triggering the UDDI Federation

When you trigger a federation, you activate the federation of objects from a source UDDI registry to one or more target registries, as defined in the federation's configuration file. If you wish to run the federation on a regular basis, for example once per hour or once daily, you can use the scheduling mechanism provided by the operating system to trigger the federation at the appropriate time intervals.

The federation has to be triggered from the machine where the CentraSite instance is located. Hence, with CentraSite as the target registry, the federation will be triggered from the target machine.

To trigger a federation, use a command of the following form:

```
CentraSiteCommand trigger Federation [-user <USER-ID>] [-password <PASSWORD>] ↩
-file <CONFIG-FILE>
```

The following table describes the complete set of input parameters that you can use with the `trigger Federation` utility:

Parameter	Description
`-user`	The user ID of a user on the source machine who has the permissions to access the required objects on the source machine.
`-password`	The password of the user identified by the parameter "-user".
`-file`	The URI (file: or http:) of the configuration file.

For example:

```
CentraSiteCommand trigger Federation [-user "AdminUser"] [-password "ABCXYZ123"] ↩
-file "C:\Federation1.xml"
```

See the section **Users and Access Rights** below for more information. If the configuration file contains the user and password information in the `sourceRegistryDetail`, you can omit these from the command line. If you supply the user and password values on the command line and also in the configuration file, the values on the command line will be used.

On Windows, you can use the Task Scheduler to trigger the federation at regular intervals. The Task Scheduler is available under Start > All Programs > Accessories > System Tools > Task Scheduler. In the Task Scheduler, supply the following values:

- The command to be activated, for example:

  ```
  C:\SoftwareAG\CentraSite\utilities\CentraSiteCommand.cmd
  ```

  If the path contains blank characters, you need to enclose the value in double quotes.
- The parameters of the command, for example

  ```
  trigger Federation [-user "AdminUser"] [-password "ABCXYZ123"] -file ↩
  "C:\Federation1.xml" ↩
  ```

- The times when the command should be activated.

On Linux and UNIX, you can use the operating system's "cron" facility to trigger the federation at regular intervals.

5 Removing a UDDI Federation from the Federation List

You can remove a federation from the list of target registries defined for federation. To do this, use a command of the following form:

```
CentraSiteCommand remove Federation [-user <USER-ID>] [-password <PASSWORD>] ↩
-name <Name>
```

The following table describes the complete set of input parameters that you can use with the `remove Federation` utility:

Parameter	Description
`-user`	The user ID of a user on the source machine who has the permissions to access the required objects on the source machine.
`-password`	The password of the user identified by the parameter "-user".
`-name`	The name of the federation.

For example:

```
CentraSiteCommand remove Federation [-user "AdminUser"] [-password "ABCXYZ123"] ↩
-name "FedName"
```

This command removes the federation defined by *Name* from the list of target registries defined for federation. The *Name* is that which you assigned when you added the federation to the federation list.

To run this command, you require permission at the operating system level to modify *emit.xml*.

6 Displaying the List of Target Registries defined for Federation

You can display the list of IDs of target registries defined for federation by using a command of the following form:

```
CentraSiteCommand list Federation [-user <USER-ID>] [-password <PASSWORD>] ↩
-name <Name>
```

The following table describes the complete set of input parameters that you can use with the `list Federation` utility:

Parameter	Description
`-user`	The user ID of a user on the target machine who has the permissions to create objects on the target machine.
`-password`	The password of the user identified by the parameter "-user".
`-name`	The name of the federation.

For example:

```
CentraSiteCommand list Federation [-user "Administrator"] [-password "manage"] -name ↩
"FedName"
```

To run this command, you require permission at the operating system level to read *emit.xml*.

7 Mapping UDDI Object IDs and tModels

This section describes how to maintain uniqueness of UDDI object IDs during the federation process, and how to map UDDI structures onto JAXR-based structures in the target registry.

Key Generator Processor

Each object in a UDDI registry has a unique ID. When an object is federated into a target registry, its ID still needs to be unique, i.e. it must not be a duplicate of an existing object in the target registry. The federation process allows you define your own keys for objects federated into CentraSite. Such keys are called *publisher assigned keys*.

The use of publisher assigned keys is activated in the federation process by using a so-called *key generator* that assigns key name domains to users who are publishers. This means that when a UDDI object is added to the CentraSite registry by a given user, the user-specific domain name is added to the UDDI object ID.

The use of key generators and publisher assigned keys is known from the UDDI standard.

The relationship between a user and the assigned domain name is maintained in a <process> element in the federation configuration file. This element has attributes that specify:

- name of user/publisher
- domain name

Example: The ID of an object in a UDDI registry is uddi:abcd1234-5678-9abc-def0-123456789abc. The domain associated with the user is "mydomain". When the object is federated to CentraSite, the object ID will be uddi:mydomain:abcd1234-5678-9abc-def0-123456789abc.

To add a key generator processor for federation, you have to modify the *emit.xml* file as follows:

To add a key generator processor for federation

1 Configure the details of the target registry in the configuration file for which you want to add the key generator processor, as described in the section **Creating a Federation Configuration File**. Execute the `add Federation` command and provide a unique identifier for this target. This will configure the target for federation and add an entry for the federation route in *emit.xml*.

2 Now open the *emit.xml* file. It will have the targets configured for federation. Locate the `<pipeline>` element that contains the attribute `id="<unique_identifier>"`, the identifier that you provided in the previous step while executing the `add Federation` command.

3 Include a `<process>` element immediately after `<pipeline id="..." >` but before `<to uri="..." />` as shown:

```
<pipeline id="<federationId>" ... >
 <process ref="KeyGeneratorProcessor_<federationId>" />
 <to uri="uddi://_<federationId>" ... />
</pipeline>
```

4 It is extremely important that you replace <federationId> with the unique identifier that you provided while executing the command. After you add the <process> element, you now have to add the following details after the <camelContext> element as shown:

```
<camelContext>
. . . .
</camelContext>

<bean id="KeyGeneratorProcessor_<federationId>"
 ↩
 class="com.softwareag.centrasite.federation.uddi.component.UddiKeyGeneratorProcessor" ↩
>
 <property name="domainKey" value="<domainKey>" />
 <property name="federationId" value="<federationId>" />
</bean>
```

5 The bean ID that you provide and the ref value in <process ref="..."> must be the same. The class name must not be modified and must be the same as provided in this example. Add the two properties <domainKey> for providing the domain name for the key generator and <federationId> to identify the federation route.

6 Now execute add Federation with the same identifier and the same target details as before. The newly added processor has to be configured for federation and this will accomplish it.

You might wish to remove the key generator processor at a later stage. To do this, proceed as follows:

To remove a key generator processor

1 Remove the <process> element and the <bean> element that you added in steps 3 and 4 above.

2 Execute the add Federation command after this so that the processor will not be used when you trigger the federation.

If the key generator processor and the taxonomy processor (see related section) are both defined, the key generator processor runs first, then the taxonomy processor runs.

Taxonomy Processor

Objects in a UDDI registry are classified according to concepts within tModels. In a registry that supports JAXR, such as CentraSite, objects are classified according to categories within taxonomies. The mapping from a UDDI classification to a JAXR-based classification uses the key value and tModel name that are contained in the `keyedReference` element of the UDDI object. The tModel name is used as the taxonomy name, and the key value is used as the category name. If the taxonomy or category do not already exist in the target registry, they are created automatically.

The concept and classification scheme are stored in tModel and keyed references of the tModel in CategoryBags of UDDI objects. The taxonomy processor processes the keyedReferences from categoryBags of the UDDI objects and creates the concepts and classification schemes in the target registry.

For example, the following are two `<keyedReference>` elements from a tModel's `<categoryBag>`

```
<keyedReference keyValue="sap.com/LMINTERNALAGENT"
   keyName=""
   tModelKey="uddi:uddi.sap.com:categorization:software-component"/>

<keyedReference keyValue="I12"
   keyName=""
   tModelKey="uddi:uddi.sap.com:categorization:business-system"/>
```

The tModelKey in the keyedReference points to a tModel. The taxonomy processor checks if this tModel can be mapped to a taxonomy in JAXR. This check is done on the basis of the UDDI to JAXR-based mapping standard. Please refer to the JAXR standard for information on how this mapping works. If the tModel can be identified as a taxonomy in JAXR according to the mapping standard, then the tModel corresponding to the tModelKey is published as a taxonomy (ClassificationScheme) in the target registry. The keyValue corresponds to concepts (or categories) in the taxonomy. So the above keyedReferences will create the following taxonomy/concept hierarchy in the target registry:

```
uddi-sap:softwareComponent
 |
 +-- sap.com
      |
      +-- LMINTERNALAGENT

uddi-sap:businessSystem
 |
 +-- I12
```

To add a taxonomy processor, you have to edit the *emit.xml* file and add a few details. Follow the instructions as given below:

To add a taxonomy processor

1 Configure the details of the target registry in the configuration file for which you want to add the taxonomy processor, as described in the section **Creating a Federation Configuration File**. Execute the `add Federation` command and provide a unique identifier for this target. This will configure the target for federation.

2 Now open the *emit.xml* file from the above mentioned location. It will have the targets configured for federation. Locate the `<pipeline>` element with the attribute id="`<unique_identifier>`", the identifier that you provided in the previous step while executing the `add Federation` command.

3 Include a <process> element immediately after `<pipeline id="..." ...>` but before `<to uri="..." />` as shown:

```
<pipeline id="<federationId>" . . . . >
 <process ref=" TaxonomyProcessor_<federationId>" />
 <to uri="uddi://_<federationId>" . . . . />
</pipeline>
```

4 In case you want to configure both the key generator and the taxonomy processor, you have to provide the key generator `<process>` element first and the taxonomy processor `<process>` element next as shown:

```
<pipeline id="<federationId>" ... >
 <process ref=" KeyGeneratorProcessor_<federationId>" />
 <process ref=" TaxonomyProcessor_<federationId>" />
 <to uri="uddi://_<federationId>" ... />
</pipeline>
```

5 It is extremely important that you replace `<federationId>` with the unique identifier that you provided while executing the command. After you add the `<process>` element, you now have to add the following details after the `<camelContext>` element as shown:

```
<camelContext>
. . . .
</camelContext>

<bean ↩
id="TaxonomyProcessor_<federationId>" class="com.softwareag.centrasite.federation.uddi.component.UddiTaxonomyProcessor" ↩
>
 <property name="csUri" value="<csUri>" />
 <!-- Ex: http://<host>:<port>/CentraSite/CentraSite -->
 <property name="federationId" value="<federationId>" />
</bean>
```

6 The bean ID that you provide and the "ref" value in <process ref="..."> must be the same. The `class=` name must not be modified and must be the same as provided in this example. Add

the two properties "csUri" for providing the Registry URL for the processor and "federationId" to identify the federation route.

7 Now execute `add Federation` with the same identifier and the same target details as before. This configures the newly added processor for federation.

You might wish to remove the taxonomy processor at a later stage. To do this, proceed as follows:

To remove a taxonomy processor

1 To remove the taxonomy processor, remove the `<process>` element and the `<bean>` element that you added in steps 3 (or 4) and 5.

2 Execute the `add Federation` command after this so that the processor will not be used when you trigger the federation.

If the key generator processor (see related section) and the taxonomy processor are both defined, the key generator processor runs first, then the taxonomy processor runs.

8 Users and Access Rights

The federation can be triggered by the a user who has the permissions needed for the source and the target registries. The target registry user specified in the configuration file must have the required permission to publish the data on the target registry, otherwise the publish operation will fail. For example, if CentraSite is the target registry, then the target registry user requires the "Asset Provider" role permission in order to publish the data.

On the target registry, the federated data will be owned by the specified target registry user. For example, if CentraSite is the target registry, and the target registry user is given as usersec1, then usersec1 will be the owner of the federated data on the target registry.

Users other than the owner can access the data on the target registry if they have the required privileges. For example, if CentraSite is the target registry, then a user having the "Asset Consumer" role permission can access the data.