



BEA AquaLogic® Enterprise Registry Repository

Exchange Utility

Version 3.0
Revised: February 2008

Contents

1. Getting Started With the ALRR Exchange Utility

What is the ALRR Exchange Utility?	1-2
Valid Metadata Entities	1-2
Example Use Cases	1-3
Related Documentation	1-3
Terminology	1-5

2. Configuring the ALRR Exchange Utility

Installing and Configuring the Exchange Utility	2-2
Install the ALRR Exchange Utility	2-2
Import the ALRR Datapack Into ALER	2-2
Verify the ALER UDDI System Settings	2-3
Configure the Service and SOA Business Entity Asset Types	2-3
Configuring the Exchange Utility Configuration File	2-5
Setting the Repository Connection Information	2-5
Setting the Registry Connection Information	2-5
Setting the Repository Query	2-6
Query by Name	2-6
Query by Registration Status	2-6
Query by Categorizations	2-6
Query by Endpoint Lifecycle Stage	2-7
Setting the Destination Registries	2-7

Setting the Registry Query	2-7
Setting the Source Registry.	2-8
Configuring ALER Categorizations In the UDDI Mappings File	2-9
Understanding the Exchange Utility’s Property File	2-9
Encrypting the Configuration File Passwords	2-10

3. Using the ALRR Exchange Utility

Running the ALRR Exchange Utility	3-2
How the Exchanged Metadata Is Synchronized.	3-4
Synchronizing the Metadata Published from ALER to ALSR	3-4
Business Entities	3-4
Endpoint.	3-4
Categorizations	3-5
WS-Policies	3-5
Registration and Active Status.	3-5
Sample Flow of Metadata from ALER to ALSR.	3-8
Synchronizing the Metadata from ALSR to ALER	3-9
Performance Metrics	3-9
Business Entities	3-10
Endpoint.	3-11
Categorizations	3-11
WS-Policies	3-11
Sample Flow of Metadata from ALSR to ALER.	3-12
Web Service Endpoint Management.	3-13
WS-Policy Management	3-14
Searching for ALSR Exchanged Metadata in ALER.	3-17
Checking the ALRR Exchange Utility Log File	3-18
Known Issues	3-19

Resynchronizing ALSR Services	3-19
Import of ALSB WSDLs.....	3-19
Workaround or Solution:	3-19

Getting Started With the ALRR Exchange Utility

This section contains information on the following subjects:

- [What is the ALRR Exchange Utility?](#)
- [Example Use Cases](#)
- [Related Documentation](#)
- [Terminology](#)

What is the ALRR Exchange Utility?

The ALRR Exchange Utility attempts to integrate ALER (AquaLogic Enterprise Repository) and ALSR (AquaLogic Service Registry) bi-directionally so that the metadata from either of these products can flow in either direction through the utility. The following are the Meta-data entities that are handled by the utility.

The ALRR Exchange Utility is capable of:

- Publishing services, endpoints, and WS-Policy artifacts from design-time to the run-time environment using UDDI.
- Submitting new services, endpoints, and policies discovered at run-time to the repository to insert a governed approval process.
- Communicating service performance information that is deposited into the UDDI registry back into the repository to better inform prospective service consumers and portfolio managers.

Valid Metadata Entities

The following metadata entities are handled by the ALRR Exchange Utility:

- All bundled Service and SCA Service types, as well any custom Services of any ALER Asset Type developed by end-users.
- Business Entities that provides Business Services. (See [Synchronizing the Metadata Published from ALER to ALSR](#) and [Synchronizing the Metadata from ALSR to ALER](#) for detailed information about how the ALRR Exchange Utility will arrive at the Business Entities.)
- Endpoint assets that are linked to the Services that provide access point to the services. For example, there can be multiple endpoints that can be tagged as *staging* or *production* and are mapped to the UDDI binding template appropriately.
- ALER Categorizations are mapped to UDDI t-models. When a categorization is applied to a Service asset, an appropriate entry is added to the UDDI Category Bag and is linked to the appropriate taxonomy t-model. These t-models are also automatically loaded into ALSR the first time they are encountered, or they can be loaded manually by the Exchange Utility.
- WS-Policy assets that are linked to either a Service or an endpoint asset are understood and are published/received based on the OASIS WS-Policy Attachment specification.

- ALSM Performance Metrics that are deposited by ALSM (AquaLogic Service Manager) into a service in ALSR are synchronized back to the endpoint asset in ALER.
- Service Registration Status and Active Status are added as items to the Category Bag of the Business Service in ALSR when the services are pushed to ALSR.

Example Use Cases

Because ALRR Exchange Utility is bi-directional, it is able to support the following use cases:

- When a services in ALSR is modified, such as when more endpoints are added to it (e.g., for staging and production environments), the ALRR Exchange Utility can relate the service with one or more endpoints in ALER because it stores a unique ID for the service in ALSR.
- When Business Services are moved to different Business Entities in ALER using asset relationships, the new relationship is also reflected in ALSR when the Services are re-synchronized, again because of the unique ID used.
- When synchronizing ALER metadata with ALSR, the ALRR Exchange Utility merges any changes so that performance metrics and other endpoint changes are preserved.
- Any two arbitrary Services that were published to ALER and ALSR can be linked using the Exchange Utility. For example, if an AquaLogic Service is published to the repository using the ALER plug-ins for Eclipse and the same Service is published to ALSR using AquaLogic Service Bus, these two services can be linked using the Exchange Utility. Once the Services are linked, they can be bi-directionally synchronized for performance metrics, Policy usage, etc.
- The endpoints of matched services can be filtered based on the specified Asset Lifecycle of the endpoints, so that only the matched endpoints are published to the repository. This query is useful when there are separate registries: one that lists the staged endpoints and another that lists the production endpoints.

Related Documentation

- [AquaLogic Service Registry 3.0](#) – a fully-compliant implementation of UDDI -v3 and a key component of SOA. The registry provides a standards based mechanism for publishing and discovering Web services and related SOA resources, such as WSDLs, XML Schemas, and XSL Transformations (XSLT).

Getting Started With the ALRR Exchange Utility

- [AquaLogic Enterprise Repository 3.0](#) – provides the tools to manage and govern the metadata for any type of software asset, from business processes and services to patterns, frameworks, applications, components, and data services.
- [ALER on dev2dev](#) – Information and resources for architects, developers, and others, including an FAQ and an overview of assembly models in Service Component Architecture (SCA).
- [SOA Governance: Essential To Your Business](#) – Learn how effective SOA governance is an essential element in any enterprise transformation strategy.
- [SOA Blog on dev2dev](#) – Keep on top of the latest SOA blogs.

Terminology

[TBD]

[Table 1-1](#) defines the terms and acronyms used this document:

Table 1-1 Terminology

Terms	Definition

Getting Started With the ALRR Exchange Utility

Configuring the ALRR Exchange Utility

This section contains the following topics:

- [Installing and Configuring the Exchange Utility](#)
- [Configuring the Exchange Utility Configuration File](#)
- [Configuring ALER Categorizations In the UDDI Mappings File](#)
- [Understanding the Exchange Utility's Property File](#)
- [Encrypting the Configuration File Passwords](#)

Installing and Configuring the Exchange Utility

Before you can use the ALRR Exchange Utility to publish and receive ALER metadata to and from ALSR, you must complete the following configuration steps.

Install the ALRR Exchange Utility

The `AlrrExchangeUtility.zip` is packaged in the `ALER3.0.1SupInteg.Doc_Data.zip` file, which is available via a hidden file when you purchase the ALER Advanced Edition. This file includes the necessary Exchange Utility files for the ALER and ALSR integration, the ALRR datapack, and the documentation.

You can unzip the `AlrrExchangeUtility.zip` into the directory on your file system where ALER is installed, typically `BEA_HOME\repository30`. When the zip file containing the ALRR Exchange Utility is unzipped to your file system, it creates the following structure.

```
alrrx <ExchangeUtility Tool Home>
|
|   datapack
|   docs
|   lib
```

Within the `<ExchangeUtility Tool Home>` directory, you will find the Exchange Utility files, such as the `alrrx.xml`, `UDDIMappings.xml`, `alrrx.properties`, `alrrx.bat`, and `encrypt.bat` files.

Import the ALRR Datapack Into ALER

The required ALRR datapack is bundled with the `AlrrExchangeUtility.zip`. Follow these steps to import the ALRR datapack (`EndpointWS.zip`) into ALER.

1. Unzip the `AlrrExchangeUtility.zip` file onto your file system.
2. Start the ALER Import/Export tool, as described in the [ALER Import/Export Guide](#).
3. Select the **Import** tab.
4. Navigate to the `<ExchangeUtility Tool Home>\datapack` directory.
5. Select the `EndpointWS.zip` as the target file to import into ALER.
6. Click **Next**, and then click **Next** again to start the import process.

7. Click **Finish** to complete the process.

Verify the ALER UDDI System Settings

Verify that the UDDI server and Web service plug-in are enabled.

1. Open the ALER **Admin** page.
2. Click the **System Settings** option in the left pane.
3. Navigate to the **External Integration > UDDI** section.
Note: You can also search for **UDDI**.
4. Verify the following UDDI property values:
 - **UDDI/cmee.uddi.enabled** is set to **True** to enable the UDDI Web service plug-in.
 - **UDDI/cmee.uddi.server enabled** is set to **True** to enable the repository to act as a UDDI registry for certain applications.
 - **cmee.uddi.business.service.relationship** is set to *BusinessService* — the relationship between Service and Business Entity asset types.
 - **cmee.import.uddi.service.assettype** is set to *Service* — the Service asset type.
 - **cmee.uddi.default.business** is set to a *UDDI Node* — only when publishing services to ALSR, when the asset is not linked to a Business Entity.
5. If necessary, click **Save**.

For more information on configuring System Settings, see the [ALER Administration Guide](#).

Configure the Service and SOA Business Entity Asset Types

Use the ALER Type Manager, enable the UDDI options for the Service and SOA Business Entity asset types, as follows:

1. Launch the Asset Editor by clicking **Edit/Manage Assets** on the Assets page.
2. Launch the Type Manager by selecting **Manage Types** on the **Actions** menu.
3. Under the Asset Types folder, select the **Service** asset type to edit its default configuration, as follows:
 - a. Change the UDDI setting to **UDDI Business Service Entity**.

Configuring the ALRR Exchange Utility

- b. Save the change by selecting **Save** on the File menu.
4. Under the Asset Types folder, select the **SOA Business Entity** asset type to edit its default configuration, as follows:
 - a. Change the UDDI setting to **UDDI Business Entity**.
 - b. Save the change by selecting **Save** on the File menu.
5. Exit the Type Manager, and then exit the Asset Editor.

For more information on using the Type Manager, see the [*ALER Registrar Guide*](#).

Configuring the Exchange Utility Configuration File

This section describes how to configure the Exchange Utility configuration file for your environment.

Setting the Repository Connection Information

Open the `alrrx.xml` file located at `<ExchangeUtility Tool Home>` and modify the following XML section so that it points to your ALER instance with the appropriate credentials.

```
<repository>
  <uri>http://localhost:7101/aler/services/FlashlineRegistry</uri>
  <credentials>
    <user>admin</user>
    <password>admin</password>
  </credentials>
</repository>
```

where URI = ALER URI, using the following format:

```
http://<host>:<port>/<aler web app name>/services/FlashlineRegistry
```

For security purposes, the password can be encrypted, as described in [Encrypting the Configuration File Passwords](#).

Setting the Registry Connection Information

The ALRR Exchange Utility can publish to one or more registry and can read from one registry. The first step is to create one more `<registry>` node with the connection info, as shown here.

```
<registries>
  <registry name="alsr">
    <inquiryURI>http://localhost:7001/registry/uddi/inquiry</inquiryURI>
    <publishURI>http://localhost:7001/registry/uddi/publishing</publishURI>
    <securityURI>http://localhost:7001/registry/uddi/security</securityURI>
    <credentials>
      <user>admin</user>
      <password>admin</password>
    </credentials>
  </registry>
  <registry name="alsr2">
    <inquiryURI>http://localhost:7201/registry/uddi/inquiry</inquiryURI>
    <publishURI>http://localhost:7201/registry/uddi/publishing</publishURI>
    <securityURI>http://localhost:7201/registry/uddi/security</securityURI>
    <credentials>
```

```
<user>admin</user>
<password>admin</password>
</credentials>
</registry>
</registries>
```

Setting the Repository Query

The following configuration snippets demonstrate how to build a query to run against ALER and receive the list of Services that should be published to ALSR. There are a number of ways that services can be queried and you can create one or more queries.

Query by Name

When the `<services>` element is configured, the service name specified is published to ALSR. However, due to a limitation in the ALER REX API, only one `<services>` element can be added.

```
<query>
  <repositoryQuery>
    <services>
      <service name="HelloWorld" />
    </services>
    <registrationStatus></registrationStatus>
    <serviceCategorizations type="AssetLifecycleStage" value="" />
    <endpointAssetLifecycleStatus></endpointAssetLifecycleStatus>
```

Query by Registration Status

When the `<registrationStatus>` element is configured, only the services with the specified Registration Status are published. For example, if this field is set to *Registered*, then only registered services will be published to ALSR, while ignoring all other matched services that are not in this state.

```
<registrationStatus>Registered</registrationStatus>
```

Query by Categorizations

When the `<serviceCategorizations>` element is configured, only the services with the specified categorization will be published. For example, when using the following categorization, only the *Recommended* services will be published to ALSR.

```
<serviceCategorizations type="classification" value="Recommended" />
```

Query by Endpoint Lifecycle Stage

When the `<endpointAssetLifecycleStatus>` element is configured, the endpoints of matched services can be filtered based on the specified Asset Lifecycle of the endpoints, and only the matched endpoints will be published to ALSR. For example, if there are two endpoints attached to a service, one with the Asset Lifecycle Stage of *Stage 3 – Build* and one with *Stage 4 – Release*, only the endpoint with the *Stage 3 – Build Asset Lifecycle* is published.

```
<endpointAssetLifecycleStatus>Stage 3 -
Build</endpointAssetLifecycleStatus>
```

This query is useful when there are separate registries: one that lists the staged endpoints and another that lists the production endpoints.

Setting the Destination Registries

The following configuration snippet demonstrates how to use the `<destinationRegistries>` element to configure one or more destination registry where the matched ALER Services will go. These registries are used when Services are picked from ALER and are moved to ALSR (i.e., ALER > ALSR).

```
<destinationRegistries>
  <destinationRegistry>alsr</destinationRegistry>
  <destinationRegistry>alsr2</destinationRegistry>
</destinationRegistries>
```

Setting the Registry Query

The following configuration snippet demonstrates how to use the `<registryQuery>` element to build a query to run against ALSR and receive the list of services that need to be fetched from ALSR and placed in ALER.

```
<registryQuery>
  <businessEntities>
    <businessEntity name="Account Services" />
  </businessEntities>
  <services>
    <service name="AddCustomerService%" />
  </services>
  <qualifiers>
```

Configuring the ALRR Exchange Utility

```
<qualifier>approximateMatch</qualifier>
</qualifiers>
<sourceRegistry>alsr</sourceRegistry>
</registryQuery>
```

Follow these configuration guidelines:

- Make sure that the `businessEntities` name or `service` name value is not empty.
- For the `businessEntities` name, the *exact* name must be specified.
- For the `service` name, at least one wildcard character should be used. For example, to get all services specify “%”.
- Search criteria for an ALSR query is case-sensitive.

Services can be searched in the following way:

- Search by one or more Service names. The Service names can be a wildcard if the qualifier is approximate. For example, if the service name is “Google”, any service that starts with “Google” will be fetched and placed in ALER.
- Select one or more Business Entity and all the services in those Business Entities will be fetched and placed in ALER. The Business Entity name has to be exact; the wild card is supported only for Services and not for Business Entities.

Warning: If both Business Entity query and Service query are specified, the Business Entity query will override the Service query.

Setting the Source Registry

The `<sourceRegistry>` element tells the registry where the Services will be picked and placed in ALER. This registry is used when Services are picked from ALSR and they move to ALER (i.e., ALSR > ALER).

```
<sourceRegistry>alsr</sourceRegistry>
```

Configuring ALER Categorizations In the UDDI Mappings File

Prior to publishing assets to ALSR, ALER Categorizations are mapped in the UDDI Mappings file (`UDDIMappings.xml`) that is stored in the `<ExchangeUtility Tool Home>` directory, as shown in the following XML snippet:

```
<uddi:uddiSettings
xmlns:uddi="http://www.bea.com/aler/integration/config/uddi">
  <categorizationMappings>
    <categorizationType alerCategorizationTypeName="AssetLifecycleStage"
alerCategorizationTypeId="112"
uddiCategoryTModelKey="uddi:bea.com:aler:categorization:AssetLifecycleStage">
      <categorization alerCategorization="Stage 1 - Propose" uddiKeyName="Stage
1 - Propose" uddiKeyValue="Stage 1 - Propose" />
      <categorization alerCategorization="Stage 2 - Plan" uddiKeyName="Stage 2
- Plan" uddiKeyValue="Stage 2 - Plan" />
      <categorization alerCategorization="Stage 3 - Build" uddiKeyName="Stage 3
- Build" uddiKeyValue="Stage 3 - Build" />
      <categorization alerCategorization="Stage 4 - Release" uddiKeyName="Stage
4 - Release" uddiKeyValue="Stage 4 - Release" />
      <categorization alerCategorization="Stage 5 - Target For Retirement"
uddiKeyName="Stage 5 - Target For Retirement" uddiKeyValue="Stage 5 - Target For
Retirement" />
      <categorization alerCategorization="Stage 6 - Retirement"
uddiKeyName="Stage 6 - Retirement" uddiKeyValue="Stage 6 - Retirement" />
    </categorizationType>
  </categorizationMappings>
</uddi:uddiSettings>
```

An ALER Categorization will be honored in ALSR only if a corresponding mapping is present in the UDDI Mappings file; otherwise, the Categorization will simply be ignored. Therefore, if a new asset Categorization is created ALER, you must regenerate the UDDI Mappings file for that Categorization to be honored in ALSR.

Understanding the Exchange Utility's Property File

This section describes the properties in the Property file (`alrrx.properties`) file that is stored in the `<ExchangeUtility Tool Home>` directory. Some properties already exist in the ALER System Settings and some of the properties are new for the ALRR Exchange Utility.

- `cmee.uddi.service.endpoint.relationship=Access` — relationship between Service and Endpoint.
- `cmee.uddi.service.wspolicy.relationship=Attached` — relationship between Service and WS-Policy and relationship between Endpoint and WS-Policy.
- `cmee.import.uddi.business.assettype=SOA - Business Entity` — Business Entity asset type.

Configuring the ALRR Exchange Utility

- `cmee.import.uddi.accesspoint.assettype=Endpoint: Web Service` — Endpoint asset type.
- `cmee.import.uddi.wspolicy.assettype=Artifact: WS-Policy` — WS-Policy asset type.
- `cmee.import.uddi.publish.ifendpointmissing=true` — If the Endpoint information is missing, this setting determines whether to publish the service or not.
- `cmee.import.uddi.artifactwsdl.relationship=Implements` — relationship between Service and WSDL artifact.
- `cmee.import.uddi.receiver.batch.size=100` — Only when receiving from ALSR, this determines the batch size.

Caution: The following properties will only be used if the corresponding property is not set in the ALER System Settings. If the ALER System Settings property is configured, that setting will override the property in the `alrrx.properties` file.

- `cmee.uddi.business.service.relationship=BusinessService` — relationship between Service and Business Entity asset types.
- `cmee.import.uddi.service.assettype=Service` — Service asset type.
- `cmee.uddi.default.business=A UDDI Node` — only when publishing services to ALSR, when the asset is not linked to a Business Entity.

For more information about other ALER System Settings, see the [ALER Administration Guide](#).

Encrypting the Configuration File Passwords

For enhanced security, the password encryption tool (`encrypt.bat`) allows you to encrypt the passwords that are stored in the Exchange Utility configuration (`alrrx.xml`) file.

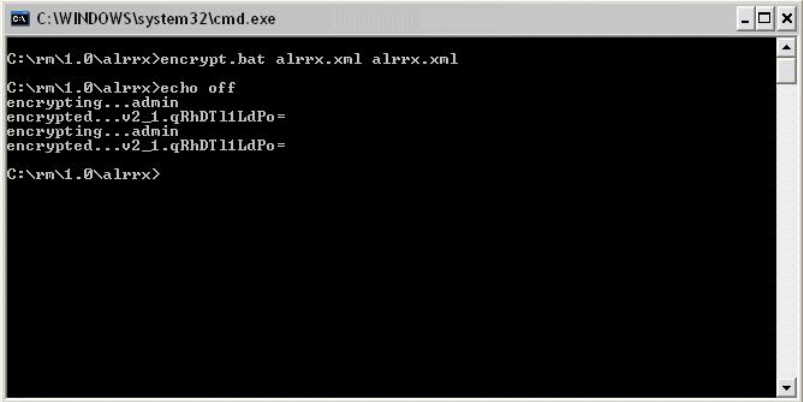
1. Navigate to the `<ExchangeUtility Tool Home>` directory.
2. From a command prompt, run the password encryption tool as follows:

```
> encrypt.bat alrrx.xml alrrx.xml
```

where:

`alrrx.xml` = the ALRR Exchange Utility configuration file

Figure 2-1 Encrypt Password Tool



```
C:\WINDOWS\system32\cmd.exe
C:\r\n\1.0\alrnx>encrypt.bat alrnx.xml alrnx.xml
C:\r\n\1.0\alrnx>echo off
encrypting...admin
encrypted...v2_1.qRhDT11LdPo=
encrypting...admin
encrypted...v2_1.qRhDT11LdPo=
C:\r\n\1.0\alrnx>
```

Configuring the ALRR Exchange Utility

Using the ALRR Exchange Utility

This section contains the following topics:

- [Running the ALRR Exchange Utility](#)
- [How the Exchanged Metadata Is Synchronized](#)
- [Searching for ALSR Exchanged Metadata in ALER](#)
- [Checking the ALRR Exchange Utility Log File](#)
- [Known Issues](#)

Running the ALRR Exchange Utility

The ALRR Exchange Utility is run from a command-prompt, using the following syntax:

```
> alrrx.bat <options>
```

[Table 3-1](#) defines the configuration options available when running the ALRR Exchange Utility.

Table 3-1 ALRR Exchange Utility Command-line Options

Option	Required	What it does...
<code>-mode <mode></code>	Yes	Configures the utility to run in either publish or receive mode. <ul style="list-style-type: none"> To publish Services to ALSR from ALER, use: <code>alrrx.bat -mode publish</code> To receive Services from ALSR into ALER, use: <code>alrrx.bat -mode receive</code>
<code>-config <File Name></code>	Optional	Passes the <code>alrrx.xml</code> file as a parameter. Example usage: <code>alrrx.bat -mode publish -config C:\rm\uddi\alrrx.xml</code> If this parameter is omitted, the configuration XML file will be loaded from the current directory where the <code>alrrx.bat</code> file is located using the system's Classpath.
<code>-map <Dir Name></code>	Optional	Generates a <code>UDDIMappings.xml</code> file by connecting to ALER and populating it with the t-models based on the configured Categorizations. Also, this loads the ALRR Exchange Utility configuration using the <code>-config</code> parameter from the default location. You can customize this file if the t-model already exists in ALSR and map an ALER categorization to a t-model. Example usage: <code>alrrx.bat -map c:/rm/uddi</code> For more information about the <code>UDDIMappings.xml</code> file, see Configuring ALER Categorizations In the UDDI Mappings File .

Table 3-1 ALRR Exchange Utility Command-line Options (Continued)

Option	Required	What it does...
-publish_tmodel <File Name>	Optional	<p>Publishes all the t-models configured in the UDDIMappings.xml file to ALSR. If a t-model already exists in ALSR, it will be skipped. You need to manually delete the existing t-models if you want the ALRR Exchange Utility to populate those t-models. Also, this loads the ALRR Exchange Utility configuration using the -config parameter from the default location.</p> <p>Example usage: alrrx.bat - publish_tmodel c:/rm/UDDIMappings.xml</p> <p>For more information about the UDDIMappings.xml file, see Configuring ALER Categorizations In the UDDI Mappings File.</p>
-link <asset_id> <service_key>	Optional	<p>Links a Service in ALER to a Service in ALSR together so that the ALRR Exchange Utility can treat them as the same service. This may be required when the Service in ALER and a service in ALSR are the same but was published to ALER and ALSR using different tools. For example, ALSM (AquaLogic SOA Management) could have published a service to ALSR and the SAM plug-in could have published the same service to ALER. After they are linked, they can be synchronized by the utility.</p> <p>Example usage: alrrx.bat -link 50822 uddi:systinet.com:demo:hr:employeesList</p> <p>Note: Use this option with caution when linking two services.</p>

How the Exchanged Metadata Is Synchronized

This section describes how metadata is synchronized when assets are exchanged between ALER and ALSR.

Synchronizing the Metadata Published from ALER to ALSR

This section describes how metadata is synchronized when publishing assets from ALER to ALSR.

Note: When synchronizing a service to ALSR that was previously synchronized, ALSR does not show the updated values if an ALSR browser instance is already open. Therefore, all the ALSR browser instances need to be restarted to see the updated values. For more information, see the [Known Issues](#).

Business Entities

Check if the Service asset being published has a related (by configured relation) Business Entity asset, as follows:

- If yes, use name of that Business Entity asset to create business in UDDI.
- If no, get default Business Entity name from configuration, as follows:
 - Check if default business entity asset is configured in the ALER System Settings.
 - If yes, use that as the default Business Entity.
 - If no, use the default Business Entity name from the `alrrx.properties` file.

Endpoint

Check if the Service asset being published has one or more related endpoint assets, as follows:

- If yes, create UDDI Binding Templates if this is the first time. If this was synchronized before, update the existing Binding Templates. Use the Endpoint URI in the Endpoint asset to arrive at the UDDI Access point. Derive the port from the WSDL that is attached to the `File Info` of the endpoint asset.
- If no, create the Binding Templates based on the WSDL that is attached to the `File Info` of the Service asset if the WSDL contains the port info.
- If Asset Life cycle is attached to the endpoint and if the Endpoint Asset Life cycle Query is used, filter the endpoints based on the Asset Lifecycle. For example, you can publish the

staging endpoint to the Staging Registry and publish the production endpoint to the production registry.

Categorizations

Check if Categorizations are applied to Service and Endpoint assets, as follows:

- If yes, load the Categorization mapping for each of the applied Categorizations from the `UDDIMappings.xml` file.
 - If the mapping is found, add an entry to the Category Bag, either to the Business Service or Binding Template.
 - If the t-model is not found in the ALSR, automatically create the t-models.
- If no, the Categorization will not be applied to the Service in ALSR.

WS-Policies

Check if Service asset being published has one or more WS-Policy assets related, as follows:

- If yes, find if these WS-Policies had been already published to ALSR before. This is detected by the t-model keys that are stored in the WS-Policy assets.
 - If the keys are found, add an entry to the UDDI Business Service Category Bag based on the WS-Policy Attachment specification.
 - If the keys are not found, create a new WS-Policy t-model on ALSR and update the Category Bag.

Registration and Active Status

The Registration and Active Status are added to Category Bags. [Figure 3-1](#) illustrates how Policy references appear in ALSR.

Figure 3-1 ALSR t-model Categories for a WSDL Service

WSDL service 'HelloWorldService'		
Categories		
tModel	key name	key value
amberpoint-com:service:serviceId	amberpoint-com:service:serviceId	uuid:AD:38E77B3
amberpoint-com:management:registeredDate	amberpoint-com:management:registeredDate	Tue Oct
systinet-com:management:state	Managed state	manage
uddi-org:wsi:types	uddi.org:wsi:types	service
uddi-org:xml:localName	uddi.org:xml:localName	HelloWo
uddi-org:xml:namespace	uddi.org:xml:namespace	http://ar
bea-com:aler:uuid	bea-com:aler:uuid	492b7165794200
http://schemas.xmlsoap.org/ws/2003/03/localpolicyreference	Associated Policy	uddi:31c32a3bc8
amberpoint-com:management:metrics:availability	Daily (percentage value)	0
amberpoint-com:management:metrics:avgResponseTime	Daily (average value in milliseconds)	0
amberpoint-com:management:metrics:faults	Daily (number)	0
amberpoint-com:management:metrics:requests	Daily (number)	0
amberpoint-com:management:metrics:availability	Weekly (percentage value)	100
amberpoint-com:management:metrics:avgResponseTime	Weekly (average value in milliseconds)	36
amberpoint-com:management:metrics:faults	Weekly (number)	9
amberpoint-com:management:metrics:requests	Weekly (number)	34
amberpoint-com:management:metrics:availability	Monthly (percentage value)	100
amberpoint-com:management:metrics:avgResponseTime	Monthly (average value in milliseconds)	69
amberpoint-com:management:metrics:faults	Monthly (number)	15
amberpoint-com:management:metrics:requests	Monthly (number)	34

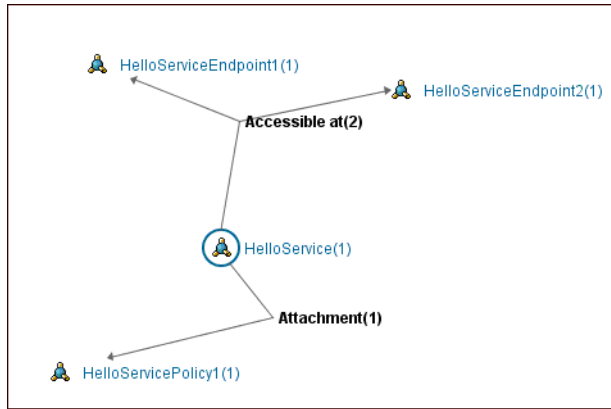
Figure 3-2 illustrates how two endpoints that are linked to a service in ALER appear in ALSR.

Figure 3-2 WSDL Bindings In ALSR

WSDL service 'HelloService'			
Bindings			
type		detail	description
access point	view	http://host1:8001	
access point	view	http://host2:8002	

Figure 3-3 illustrates how the different entities and their relationship show up in the ALER Navigator.

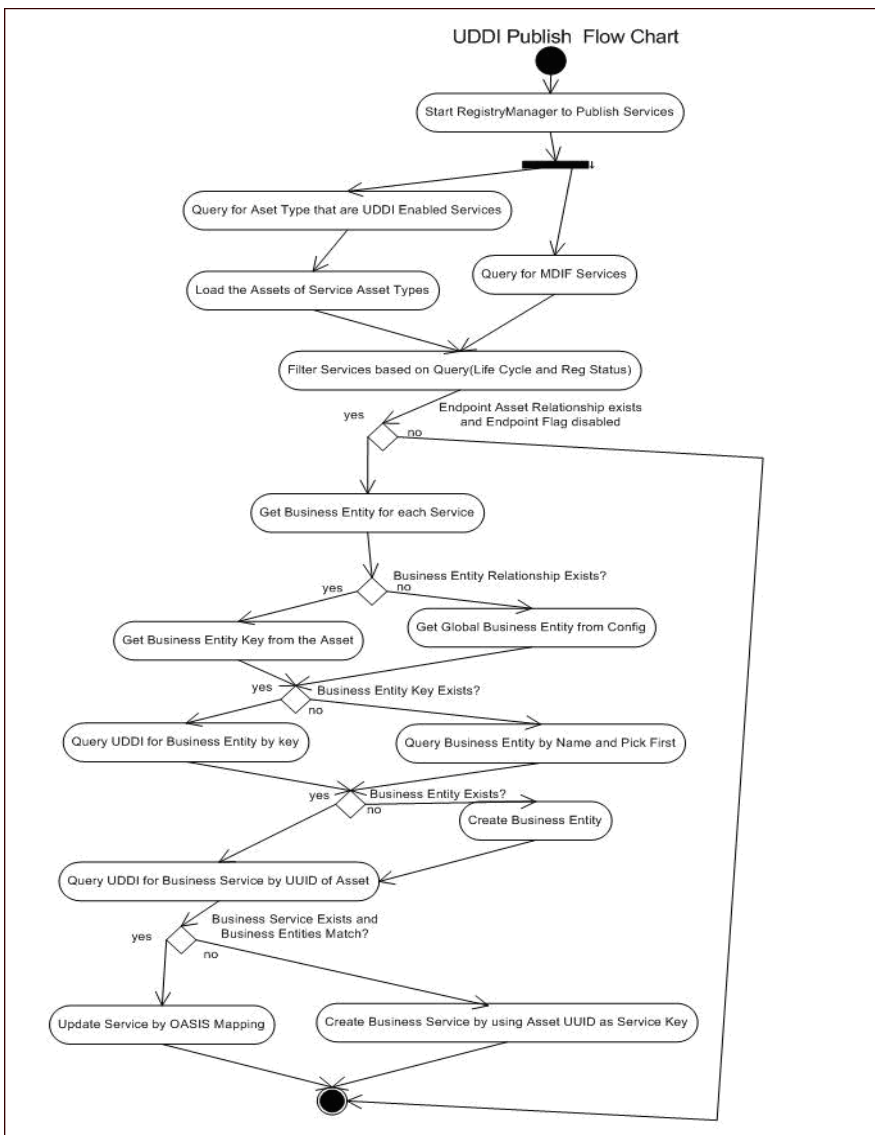
Figure 3-3 Entity Relationship in ALER Navigator



Sample Flow of Metadata from ALER to ALSR

Figure 3-4 illustrates the ALER > ALSR metadata synchronization described in this section.

Figure 3-4 Flow of Metadata Published from ALER to ALSR



Synchronizing the Metadata from ALSR to ALER

This section describes how metadata is synchronized when receiving assets into the repository from ALSR.

Performance Metrics

ALRR Exchange Utility updates the endpoint asset with the performance metrics deposited by ALSM. The `UDDIMappings.xml` file contains the mapping between the performance metrics t-models and the keys to ALER custom fields where these metrics are populated. You can also customize the UDDI Mapping file so that the metrics can appear on any tab mapped to any field in the tabs of any custom asset type.

Figure 3-5 illustrates how the performance metrics deposited by ALSM shows up in ALER after the endpoint is synchronized by the Exchange Utility.

Figure 3-5 ALSR Operational Information

Operational Information	
Availability - Daily (%) :	100
Availability - Weekly (%) :	100
Availability - Monthly (%) :	100
Daily Average Response Time (milliseconds) :	136
Weekly Average Response Time (milliseconds) :	37
Monthly Average Response Time (milliseconds) :	77
Daily Requests :	16
Weekly Requests :	30
Monthly Requests :	30
Daily Faults :	1
Weekly Faults :	12
Monthly Faults :	14
Start Date for Metrics Monitoring :	Tue Oct 02 14:51:48 PDT 2007
Last Updated :	Fri Oct 19 09:47:44 PDT 2007

Figure 3-6 illustrates how the performance metrics deposited by ALSM appear in ALSR.

Figure 3-6 ALSR Performance Metrics for an ALSM WSDL Service

WSDL service 'HelloWorldService'		
Categories		
tModel	key name	key value
amberpoint-com:service:serviceId	amberpoint-com:service:serviceId	uuid:AD:38E77B3
amberpoint-com:management:registeredDate	amberpoint-com:management:registeredDate	Tue Oct
systinet-com:management:state	Managed state	manage
uddi-org:wsdl:types	uddi.org:wsdl:types	service
uddi-org:xml:localName	uddi.org:xml:localName	HelloWo
uddi-org:xml:namespace	uddi.org:xml:namespace	http://ar
bea-com:aler:uuid	bea-com:aler:uuid	492b7165794200
http://schemas.xmlsoap.org/ws/2003/03/localpolicyreference	Associated Policy	uddi:31c32a3bc8
amberpoint-com:management:metrics:availability	Daily (percentage value)	0
amberpoint-com:management:metrics:avgResponseTime	Daily (average value in milliseconds)	0
amberpoint-com:management:metrics:faults	Daily (number)	0
amberpoint-com:management:metrics:requests	Daily (number)	0
amberpoint-com:management:metrics:availability	Weekly (percentage value)	100
amberpoint-com:management:metrics:avgResponseTime	Weekly (average value in milliseconds)	36
amberpoint-com:management:metrics:faults	Weekly (number)	9
amberpoint-com:management:metrics:requests	Weekly (number)	34
amberpoint-com:management:metrics:availability	Monthly (percentage value)	100
amberpoint-com:management:metrics:avgResponseTime	Monthly (average value in milliseconds)	69
amberpoint-com:management:metrics:faults	Monthly (number)	15
amberpoint-com:management:metrics:requests	Monthly (number)	34

Business Entities

Check if the Service asset being received exists and is related to a Business Entity asset, as follows:

- If yes, the same Business Entity relationship will be used.
- If no, the Business Entity on the ALSR side will be used. If the Business Entity does not exist in ALER, it will be created.

If the Service asset is newly created, get the default Business Entity asset type for UDDI Business from the ALER configuration:

- If found in the System Settings
- If not, from the `alrrx.properties` file

Check if an asset exists with that name and type, as follows:

- If yes, simply relate that existing asset to newly created service asset using configured relation.
- If no, create a new asset and relate it to newly created service asset using configured relation.

Endpoint

Check if the Service asset being received has one or more Endpoint assets related, as follows:

- If yes, create check if the Endpoint assets are the same as the existing Binding Templates using the UUID. If they are same, update the Endpoints.
- If no, create the Endpoints for each Binding Template and relate them to the Service asset.

Categorizations

Check if Categorizations are present in the Category Bag, as follows:

- If yes, load the Categorization mapping for each of the applied Categorizations from the `UDDIMappings.xml` file.
- If the mapping is found, set the Categorizations of the Service asset.

WS-Policies

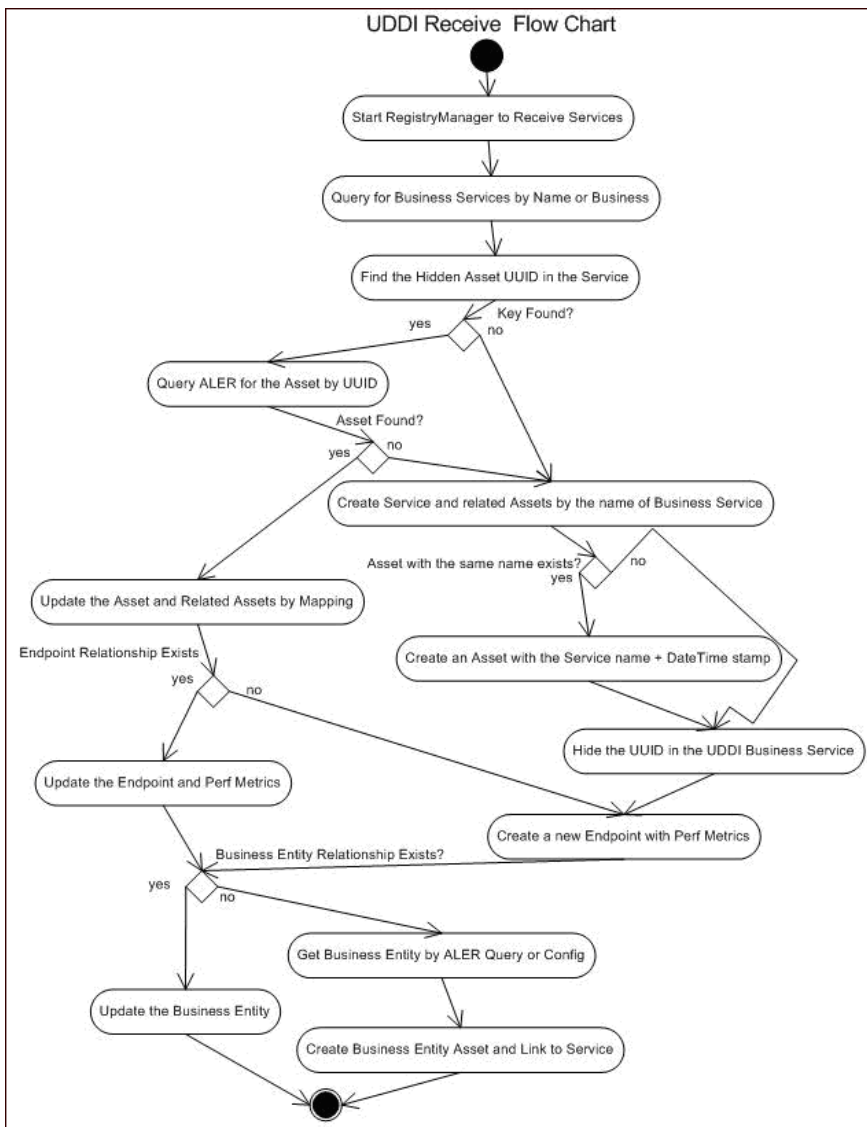
Check if the Service asset being received has one or more WS-Policy assets related, as follows:

- If yes, check if these WS-Policies are the same as the Policies that are applied to the UDDI Service. This is detected by the t-model keys that are stored in the WS-Policy assets.
 - If the keys are found and match, update the existing WS-Policy assets.
 - If the keys are not found, create a new WS-Policy asset and related the Service asset with the newly created asset.

Sample Flow of Metadata from ALSR to ALER

Figure 3-7 illustrates the ALSR > ALER metadata synchronization described in this section.

Figure 3-7 Flow of Metadata Received from ALSR



Web Service Endpoint Management

Use the ALER Asset Editor to add the new `Endpoint:Web Service` asset type so that the endpoint information can be published to ALSR.

1. From the File menu, select **New** to create a new `Endpoint:Web Service` asset type.
2. Add a description if necessary.
3. Select the **Taxonomy** tab to relate the asset to the Service asset by:
 - a. Scroll down to the Relationships section and click **Add**.
 - b. On the Add Relationship dialog box, select the **Provides Access to** Relationship Type.
 - c. Click **OK**.
 - d. Click **Approve** to save the change.
4. Select the **Overview** tab, and do the following:
 - a. In the Endpoint URI field, click **Add** to enter the **URI** for the asset. Then click **OK** to add the Endpoint URI.
 - b. In the File Information field, click **Add** to enter the effective **WSDL** that contains the port. Then click **OK** to add the WSDL.

Note: You can omit this step if the WSDL attached to the Service contains the port information.
 - c. Click **Approve** to save the change.

Figure 3-8 Configure EndPoint:Web Service Asset Type

HelloWorldServiceEndpoint (1)
Asset Type : Endpoint: Web Service

Support Metrics Change Management Management Review Miscellaneous Administration
Overview Taxonomy Technical Operational Information Documentation Tests

Name HelloWorldServiceEndpoint
Version 1
Description
Endpoint URI http://mpalanis01.amer.bea.com:7021/tutorial/HelloWorldService Edit
Producing Project(s)
Add Delete
Community
File Information
Name Description URL Add
WSDL Location WSDL location URI http://mpalanis01.amer.bea.com:... Edit Delete

5. Optionally, you can set the Asset Lifecycle to this asset if the Endpoint needs to be filtered by the Asset Lifecycle.

For more information on using the Asset Editor to manage assets, see the [ALRR Registrar Guide](#).

WS-Policy Management

Use the ALRR Asset Editor to add a new WS-Policy so that the endpoint information can be published to ALSR.

1. From the File menu, select **New** to create a new `Artifact:WS-Policy` asset type.
2. Add a description if necessary.
3. Select the **Taxonomy** tab to relate the asset to the Service asset or Endpoint:

- a. Scroll down to the Relationships section and click **Add**.
 - b. On the Add Relationship dialog box, select the **Attaches to** Relationship Type.
 - c. Click **OK**.
 - d. Click **Approve** to save the change.
4. Set the WS-Policy document that contains the Policy statements.
 5. Select the **Overview** tab, and do the following:
 - a. In the File Information field, click **Add** to enter the location of the WS-Policy document that contains the Policy statements. Then click **OK** to add the WS-Policy location.
 - b. Click **Approve** to save the change.

Figure 3-9 Configure Artifact:WS-Policy Asset Type

Logging, Message History (2) (1)
Asset Type : Artifact: WS-Policy

Support Metrics Change Management Management Review Miscellaneous Administration
Overview Taxonomy Technical Operational Information Documentation Tests

Name: Logging, Message History (2)
Version: 1
Description: Logging, Message History (2)

Producing Project(s):
Add Delete

Community:
Packaging Description:

File Information

Name	Description	URL
Policy Location	Policy location URI	http://mpalanis01.amer.bea.com:...

Add Edit Delete

Using the ALRR Exchange Utility

For more information on using the Asset Editor to manage assets, see the [ALER Registrar Guide](#).

Searching for ALSR Exchanged Metadata in ALER

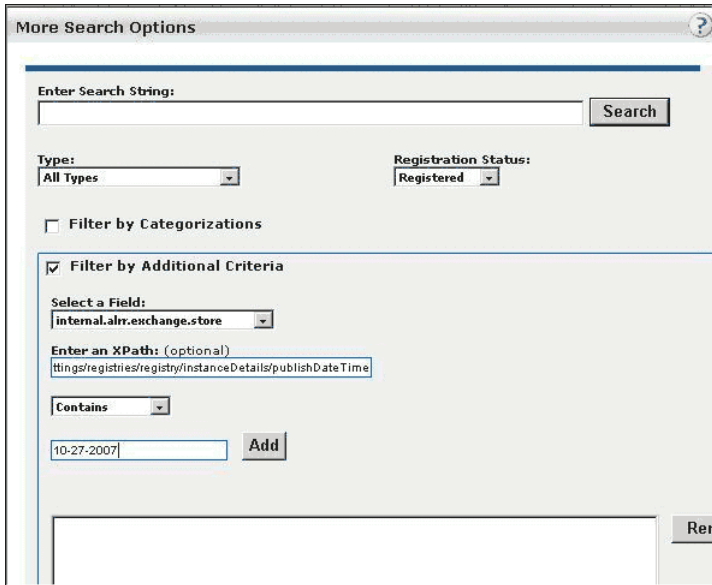
The ALRR Exchange Utility tags each published and received Service with information that can be used for querying, as follows:

- Publish or Receive Date and Time
- Source and Destination Registries.

To query published/received Service information, use the ALER More Search Options feature, as follows:

1. In the ALER Web console, open the **Assets** page.
2. In the Search box in the sidebar, click the **More Search Options** link.
The More Search Options dialog box opens.
3. Select the **Filter by Additional Criteria** check box to reveal the additional filtering criteria options.
4. On the Select a Field drop-down, select the **internal.alrr.exchange.store** option.
5. In the field next to the Add button, enter the date the metadata was exchanged.

Figure 3-10



6. Click the **Search** button at the bottom of the dialog box.

For more information on using the ALER search options, see the [ALER User Guide](#).

Checking the ALRR Exchange Utility Log File

The ALRR Exchange Utility uses `log4j` for logging the detailed tasks performed. The log file is stored in the `<ExchangeUtility Tool Home>` directory. The logging options can be changed by updating the file `log4fl.properties` file located in the `<ExchangeUtility Tool Home>` directory.

Known Issues

This section describes the known issues when using the ALRR Exchange Utility.

Resynchronizing ALSR Services

When synchronizing a Service to ALSR that was previously synchronized, there is known issue where ALSR does not show the updated values if an ALSR browser instance is already open. Therefore, all the ALSR browser instances need to be closed to see the updated values.

Import of ALSB WSDLs

ALER WSDL import is not currently capable of supporting the import of an XSD into a WSDL document using the WSDL import mechanism. This is considered improper use of the WSDL import element by the industry. An example of the improper usage of the WSDL import element to import an XSD, along with an example of the correct way to import XSD into a WSDL is included. Note that ALER WSDL import does support the importing of WSDL into a WSDL document using the WSDL import element.

ALSB (AquaLogic Service Bus) currently generates WSDLs that incorrectly import XSD using the WSDL import element. This causes a problem in the AL suite due to the fact that these ALSB WSDLs can be parsed and submitted to ALER properly by the AL common Eclipse tooling; however, the ALRR Exchange Utility is not capable of parsing the WSDLs when migrating them back to ALSR (AquaLogic Service Registry).

Workaround or Solution:

Example of improper usage of the WSDL import element to import XSD:

```
<?xml version='1.0' encoding='UTF-8'?>

<definitions name='OrderProcessing'
targetNamespace='http://avitek.com/orderprocessing/definitions'
xmlns:tns='http://avitek.com/orderprocessing/definitions'
xmlns:po='urn:iwaysoftware:ibse:jul2003:createPO'
xmlns:por='urn:iwaysoftware:ibse:jul2003:createPO:response'
xmlns:pos='urn:iwaysoftware:ibse:jul2003:POstatus'
xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
xmlns='http://schemas.xmlsoap.org/wsdl/'>

  <import namespace='urn:iwaysoftware:ibse:jul2003:createPO'
location='bapipo.xsd' />

  <import namespace='urn:iwaysoftware:ibse:jul2003:createPO:response'
```

Using the ALRR Exchange Utility

```
location='bapipor.xsd'/>
  <import namespace='urn:iwaysoftware:ibse:jul2003:POStatus'
location='POStatus.xsd'/> .
```

Example of proper usage of the XSD import element to import XSD:

```
<?xml version='1.0' ?>
<wsdl:definitions targetNamespace='urn:listing2'
  xmlns:tns='urn:listing2'
  xmlns:listing3='urn:listing3'
  xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
  xmlns:xsd='http://www.w3.org/2001/XMLSchema'
  xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'>
  <wsdl:types>
    <xsd:schema targetNamespace='urn:listing2'
      xmlns:listing3='urn:listing3'
      xmlns:xsd='http://www.w3.org/2001/XMLSchema'>
      <xsd:import namespace='urn:listing3'
        schemaLocation='listing3.xsd' />
```