Best Practices for Deploying AquaLogic Service Bus™ Resources

Version: 2.6

Document Date: January 2007

Best Practices for Deploying AquaLogic Service Bus Resources

AquaLogic Service Bus 2.6 has powerful capabilities to automate the deployment of resources to production systems and also provides complete support for various deployment use cases.

This document describes the best practices and procedures that may be used while deploying AquaLogic Service Bus 2.6. It is targeted towards deployers or administrators who plan out the deployment and topology rather than developers who write automated scripts or programs for deployment. For more information, see Using the Deployment APIs in the *Deployment Guide*. To write the scripts or programs, developers can refer to AquaLogic Service Bus Javadocs located at http://e-docs.bea.com/alsb/docs26/javadoc/.

Deployment Topologies

The following sections describe typical deployment topologies.

Development and QA Systems

There may be multiple development systems for a given production system. For example, each department for an enterprise ESB production system might have a separate project with a development system for that project's team. The development system typically has a QA system associated with it. The two systems are related one is to one.

Stage and Production Systems

Typically, there is a stage system associated with the production system. These two systems are related one is to one. The stage system mirrors the production system as much as possible. Typically, the relationship of the development system (or QA system) to production systems (or stage systems) is many is to one.

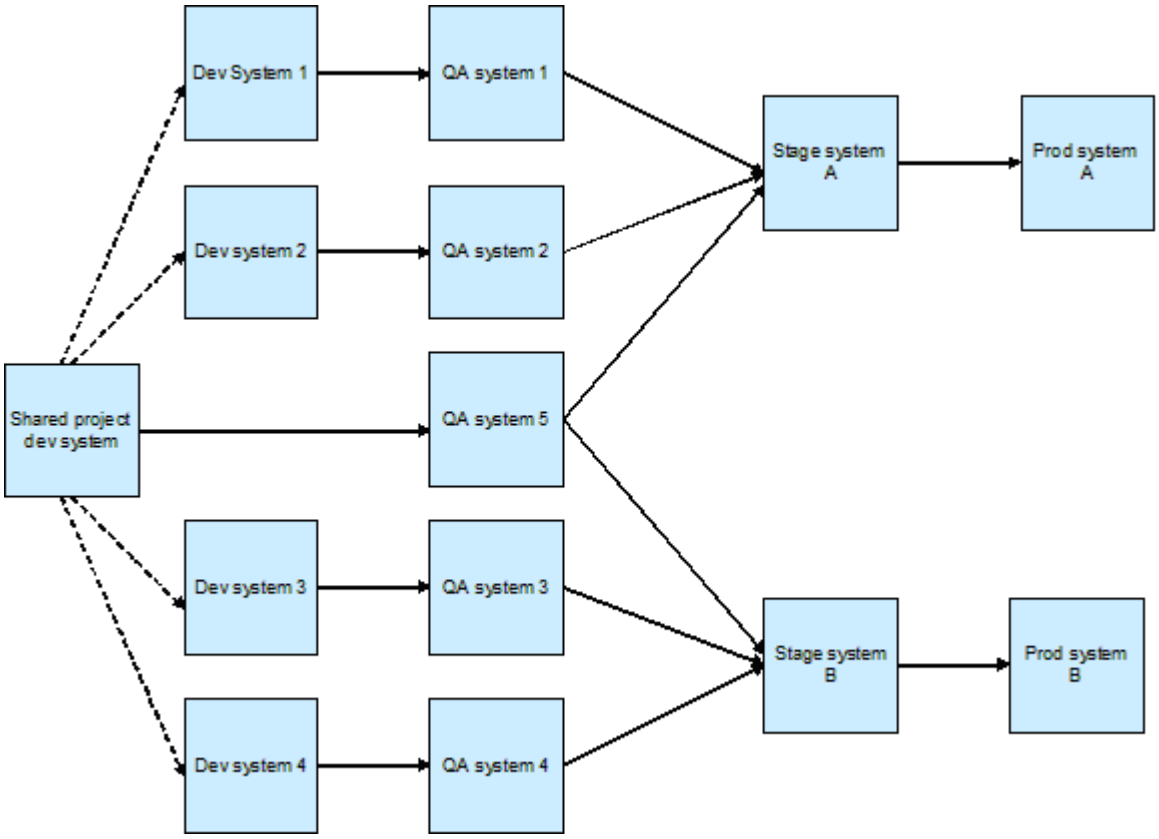
Shared Projects

In more complex deployment topologies (a development system for each department in a large organization) corporate-wide standards like schemas, transformation and shared services may be in a shared project. This scenario can be handled by having a separate development system for the shared project and providing a read-only copy of the shared project in each departmental development system. When any shared resource changes, all the departmental systems are updated by importing and exporting the required resources.

When deploying resources into a stage system or a production system, the shared project along with all the impacted departmental projects should be deployed together to avoid any semantic errors that might cause the deployment to fail.

Sometimes, a shared project may be shared across multiple production systems. For example, when there are separate production systems for each department that are interconnected into a corporate ESB network, corporate-wide assets like schemas may be shared across all the production systems.

Complex Topology (complex_topology.gif)



Types of Project Deployment

The typical unit of deployment is called a project. There are two types of project deployment:

- **Complete** - The entire project is deployed
- **Incremental** - Only changes to the project since the last deployment or some explicitly selected subset of resources are deployed.

  Incremental deployment is additive – only new and updated resources are deployed on the target system. Note that any resource that is deleted in the project is not deleted from the target system. However, deleted project resources are also deleted from the target system during a complete project deployment.

  Note: This behavior may be customized at import time. For example, alert destinations are defined in the production system by the production system operator, but these alerts do not exist in the development system. In such a scenario, you may not want to delete these alert destinations when you do a complete deployment of the project.

This document focuses on these basic types of deployment. Use cases involving shared projects are extensions of the basic use cases. In addition, this document also focuses on the use cases where export and import of resources is automated by scripts or programs.

Deployment Roles

Export, import and environmental customizations may be done by a deployer, operator or Administrator depending on the system and depending on the enterprise's policy.

Export from the development system is typically done by a deployer. The Administrator, operator, or deployer may be responsible for exporting and importing resources from the stage system to the production system.

Exporting and importing resources can be done using AquaLogic Service Bus Console or via a script or program (which can be written by a developer).

If an operator is responsible for exporting resources from a system, a pre-defined automated script or program can be executed to export either the complete project or specific resources in the project. Similarly, if an operator is responsible for importing resources into a system, a pre-defined automated script or program can be executed to do the import.

Customization Files

An Administrator uses customization files to make changes to environment values as well as to change references within resources. Customization files can include customizations for all the environment values found in the selected resources, including complex environment values types defined inside the `EnvValueTypes` class. In addition, it includes a reference customization type for changing resource references inside resources with dependencies.

The customization schema (`Customization.xsd`) which describes the customization types is available at the following location in your AquaLogic Service Bus installation:

```
BEA_HOME\weblogic92\servicebus\lib\sb-public.jar
```

You can create sample customization files from AquaLogic Service Bus Console. The scope of a customization file can be a project or individual resources in a project. For more information, see "Creating Customization Files" in System Administration in *Using the AquaLogic Service Bus Console.*

The created sample customization file may be used as a starting point for making desired modifications by specifying the actual values for an environment during the export or import process.

Substituting Environment Values

As part of deployment, environment values in resources in source systems must be changed as part of the export process or the import process to reflect the values that are application in the target system.

Environment values are certain pre-defined fields in the configuration data whose values are very likely to change when you move your configuration from one domain to another (for example, from test to production). Environment values represent entities such as URLs, URIs, file and directory names, server names, e-mails, and such. Also, environment values can be found in Alert Destinations, proxy services, business services, SMTP Server and JNDI Provider resources, and UDDI Registry entries.

Certain environment values are complex XML objects that cannot be found and replaced using the Find and Replace option from AquaLogic Service Bus Console. However, you can still set these environment values directly by using the ALSBConfigurationMBean from a script. For detailed information about ALSBConfigurationMBean, see the Javadocs for AquaLogic Service Bus Classes. In addition to setting them through the API, you can set complex type environment values using customization files.

Customization files are XML files and you can open these files in any editor and substitute the required environment values. In addition, you can search for specific environment values (that are

not complex XML types) in AquaLogic Service Bus Console or in a customization file and replace them with the new values. You can fine-tune the scope of the search by filtering these environment values based on variable type or project. For more information, see "Finding and Replacing Environment Values" in System Administration in *Using the AquaLogic Service Bus Console*.

Note: After you substitute the environment values, you must execute these customization files. For more information, see "Executing Customization Files" in System Administration in *Using the AquaLogic Service Bus Console*.

You can substitute environment values as part of the import or export process.

Exporting Resources

The exporter needs to know if incremental deployment is feasible. If so, a developer can write a script that can find all resources that have been changed or created after a specific time frame (for example, after the last deployment) and export them.

If the exporter uses AquaLogic Service Bus Console for exporting only selected resources (for incremental deployment), the exporter can look at the last modified timestamp of the resources and select the only those resources that need to be exported. For more information, see "Exporting Resources" in System Administration in *Using the AquaLogic Service Bus Console*.

Customizing Resources During Export

You can customize resources (including substituting environment values) while importing or exporting resources. When you export resources:

1. Start a session.
2. Customize the resources and execute the customization file
3. Export the required resources.
4. Discard the session to ensure that the original values are retained at the source system. Since the exported file contains the updated values, these values are reflected at the target system.

Importing Resources

You can import resources from AquaLogic Service Bus Console or using a script. For information about importing resources using the console, see "Importing Resources" in System Administration in *Using the AquaLogic Service Bus Console*.

When a script is used, the script stages the import JAR. It then obtains a deployment plan from the system for the staged import JAR. The deployment plan should contain all resources in the project and if these resources should be created, deleted, updated or left untouched in the target system. The deployment plan is typically customized by the script to not overlay resources that are environment specific (like service account) or overlay or delete operator controlled resources.

For more information, see "Importing Resources" in System Administration in *Using the AquaLogic Service Bus Console*.

Customizing Resources During Import

You can customize resources (including substituting environment values) while importing or exporting resources. When you import resources:

1. Start a session.
2. Import the required resources.
3. Customize the resources and execute the customization files.
4. Activate the session.

On import, you can preserve environment variables and ensure that only environment values of resources that are created on import are customized by the customization file.

Importing Environment-specific Resources

Resources like service accounts, proxy service providers, SMTP server, JNDI server, and UDDI server are fundamentally environment specific and under the control of administrators. You need to define these resources in the target system and avoid moving such resources across environments

Even in a complete project deployment, such resources should not be updated in the target system. If there is a reference in the import file to an environment-specific resource that does not exist on the target system, the suggested best practice is to fail the import and redo the import after the administrator defines these resources.

Sometimes, there might be cases where the name of the same environment resource (like an SMTP server) is different across environments. This can be handled by the import script by mapping the reference to such a resource as part of the import process.

Importing Operational Values

There are two types of operational values in AquaLogic Service Bus, global Operational Settings and operational values for proxy and business services. Global Operational Settings is a resource located under the System project folder. The global Operational Settings resource can be exported like any other resource. On import, this resource can only be updated; it cannot be created or deleted.

On import, you can preserve the operational values in the target system by selecting the **Preserve Operational Values** option in AquaLogic Service Bus Console.

Note: Sometimes, operators define alert destinations for SLA alerts. Such resources are operator controlled and should not be overlaid or deleted during deployment. On import, SLA alert rules are not preserved, they are merged. For example, if the importing domain has SLA alert rules A, B, and C for a service and the import JAR has alert rules C and D. Rules A and B remain intact. Rule D gets added, and rule C gets overwritten.

Deploying WebLogic Server Artifacts

Ideally, deployment should fail if WebLogic Server artifacts required for completing the configuration are missing.

While some are checked during import, others are not checked (for example, request queue for a business service) and a runtime error occurs.

The best practice is to have the deployment script retrieve environment values for deployed services and alert destinations and check with WebLogic Server to see if the specified resource exists in WebLogic Server. If not the deployment fails and the administrator has to define these resources before the script is re-executed by the operator.

A more limited alternative is to import the artifacts into a session and look for semantic errors. Semantic errors occur if AquaLogic Service Bus checks for the existence of a WebLogic Server resource at design time.

Summary of Deployment Best Practices

This is a summary of BEA recommended best practices for deployment of AquaLogic Service Bus resources.

- Avoid project renames. If projects need to be renamed, do so concurrently across development, QA, stage and production systems before the next deployment.
- Operations on environment specific resources are best filtered out at import time. Administrators may define these resources (referenced in the import file) in the target system before starting the import process. Alternatively, new environment-specific resources may be deployed and customized for the environment after the import.
- Operational resources controlled by the operator should not be impacted by the import. A naming convention or a dedicated folder could be used to identify such resources during import.
- When the customization file contains the customizations of all project resources in a single file, apply customizations only to resources that are imported. An alternative is to preserve environment values on import and only apply customizations to resources that are added during import.
- At export-time, the exporter needs to know if resources have been deleted, renamed, or moved; and if there have been project reorganizations of artifacts since the last export. If yes, complete deployment should be done. If no, the exporter can select only those resources that need to be exported and an incremental deployment can be done.

UDDI

There are special considerations to factor in when working with services deployed in UDDI registries. This section provides information about the proxy services and business services that are published to or imported from UDDI. The use of UDDI production registries complicates import into the production system and special care has to be taken.

UDDI Deployment Topologies

Based on your environment, you can have any of the following UDDI deployment topologies:

- Development-only registry
- Production-only registry
- Development and production time registry
- Registry per individual domain

Development-Only Registry

The simplest deployment of UDDI is having a single development (design) time registry where the resources are both published and discovered. This registry is used for governance using approval control. However, you can also have separate design time publish registry and discovery registry. After the design time publish registry is approved, it can be promoted to the discovery registry.

Production-Only Registry

In this production time registry topology, a single production UDDI registry contains all the production business services and proxy services, and their locations (URL). The production registry can be used to discover all production services. However, you can also have separate production time publish registry and discovery registry. After the production time publish registry is approved, it can be promoted to the production discovery registry.

A key reason for having a production registry is dynamic endpoint management. You can change the business service location in the UDDI registry and have it automatically reflected in the AquaLogic Service Bus production system by enabling automatic synchronization of resources. For more information, see "Auto-Synchronization of Services With UDDI" in UDDI in *AquaLogic Service Bus User Guide*.

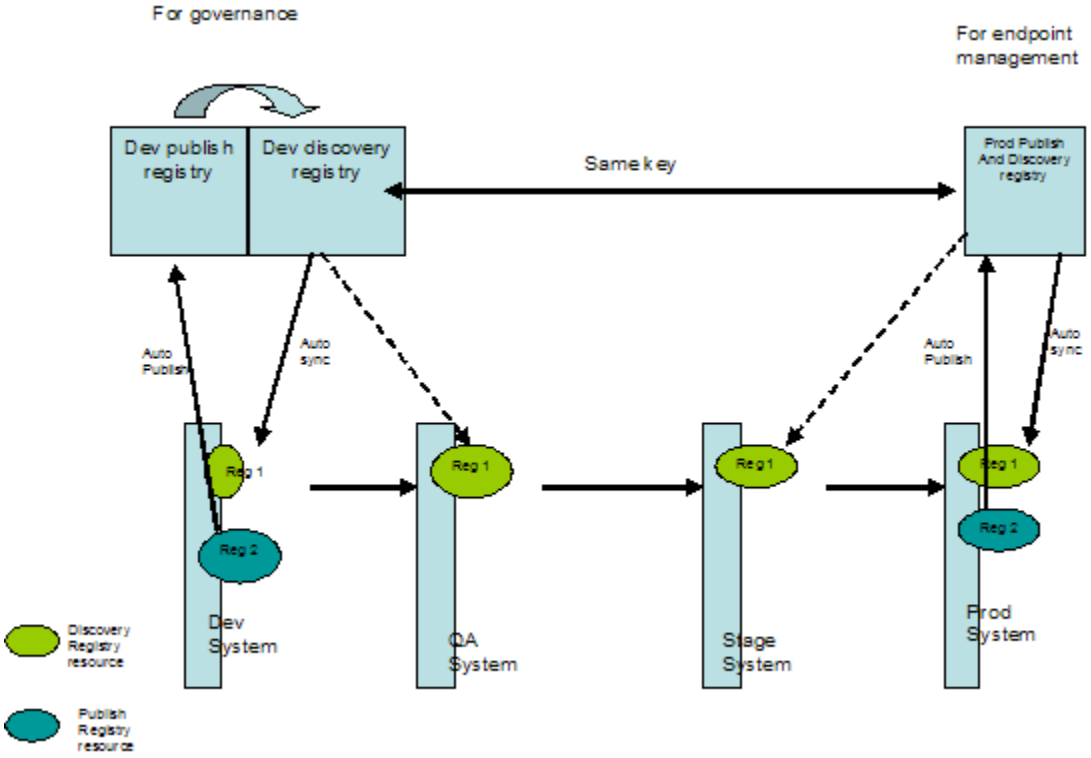
Typically, proxy services are automatically published to the publish registry. If the approval step in UDDI results in a reject of the service, the UDDI approver manually notifies the developer and the developer has to make the appropriate changes and re-export the services through the stage and production systems. For more information, see "Auto Publish" in UDDI in *AquaLogic Service Bus User Guide*.

Production business services are imported from the production discovery registry. Automatic synchronization of the resources can be enabled to ensure that changes in the UDDI registry are automatically updated in AquaLogic Service Bus. However, when you manually import business services from the production registry into the development registry, you must change the environment values to conform to the values in the development system. The development business service is not automatically synchronized with the UDDI registry because the development environment values would most probably be different from the production registry.

Development and Production Registry

An alternative approach is to have a separate development and production UDDI registry. In this topology, the approval step can take place in the development registry. So, the extra development UDDI registry ensures that approvals are done earlier in the cycle.

Complex Topology (complex_topology_uddi.gif)



Registry per Individual Domain

Sometimes, the proxy service does a dynamic search of the UDDI registry with a POJO callout to select the service meeting the desired search criteria and retrieve the URL for dynamic routing. In this scenario, a dummy service is defined in AquaLogic Service Bus with a dummy URL and the URL is dynamically replaced with the actual value after the lookup. In this scenario, a UDDI registry is needed for each environment. Also, the dummy business service in AquaLogic Service Bus is not linked to any UDDI service.

There is an impact on performance and availability in this scenario. So, the preferred approach is to enable automatic synchronization of a business service into AquaLogic Service Bus with a routing table XQuery resource that can be used for dynamic routing. For more information, see "Using Dynamic Routing" in Modeling Message Flow in AquaLogic Service Bus in *User Guide.*

Summary of UDDI Deployment Best Practices

This is a summary of BEA recommended best practices for working with services deployed in UDDI registries.

- Organize business services associated with a particular UDDI registry in a specific folder to make it easy to identify these resources during import.
- Use the same UDDI server resource name in all systems using that UDDI registry. When there is a separate development and production UDDI registry, use the same resource name for the development and production instances of the UDDI registry. This ensures that references to the server by services are automatically resolved during import.

- Preserve the same service key across the two registries when you use a development and production UDDI registry.

  Typically, the automatic publish option is enabled for production proxy services and the automatic synchronization option is enabled for business services. When the service is manually imported from the UDDI registry into the development system, the service still has information about the UDDI key of the source registry and a reference to the registry resource. The development service is imported into stage or QA systems in the normal way. The service in the stage and QA systems also has the UDDI key and registry reference to the source registry of the service. So, ignore any UDDI-related status messages in these systems that prompt you that the service is out of sync or that the service should to be deleted.

  The same service in development and production may not have the same keys when two registries are used. However, the recommended approach is to preserve the same keys.

- Create a new UDDI service with the new shape when the shape of a business service changes.

Importing and Exporting Resources Between Multiple Systems

You can import or export UDDI related resources available in AquaLogic Service Bus from one system to another system. This section provides information you need to take into account in the various UDDI deployment topologies. In all topologies, all systems have the same UDDI registry resources configured. However, auto synchronization is not enabled in the stage and QA systems.

There are a few factors to consider when a production UDDI registry is used:

- UDDI contains only a subset of the information for a service. So, some part of the information related to the service comes to the production system through the normal import export process while other parts of the service information comes into the production system from synchronization with the UDDI registry. This information has to be carefully managed during import.
- Ideally, when a service comes into the development system from a design time UDDI registry, there has to be some way to identify an equivalent service in the production UDDI registry. This ensures that at import time, the information from that service can be merged into the equivalent service in the production UDDI registry.

Development-Only registry

In this scenario, when you move one system to another system, you need to do the following:

1. Detach the business service from the UDDI registry.
2. Disable the auto publish option for the proxy service. You can do this using the MBean API or do this from AquaLogic Service Bus Console while configuring the proxy service.
3. If required, make any changes to the service.
4. Manually export the resource from the current system to a JAR and import it to the next system.

Production-Only and Development & Production Registry

When importing resources into the production system, there can be any one of the following scenarios:

- Importing a new service into a system, and the UDDI key of the service in the import JAR and the UDDI registry are the same.
- Importing a new service into a system, and the UDDI key of the service in the import JAR and the UDDI registry are the different.
- Importing an existing service into a system

Importing a new service into a production system, and the UDDI key of the service in the import JAR and the UDDI registry are the same

In this scenario, import the service into the system and do one of the following:

- Use the import script to explicitly force a UDDI re-synchronization of the service in the current session with an API call.
- Mark the imported service for background re-synchronization by AquaLogic Service Bus.

  The disadvantage of this option is that UDDI-related service fields may be incorrect until you complete the re-synchronization. However, if auto synchronization is enabled, this is a small interval.

Importing a new service into a production system, and the UDDI key of the service in the import JAR and the UDDI registry are the different

In this scenario, import the service into the system and do one of the following:

- Use the import script to explicitly force a detach of the service from the UDDI registry with an API call in the current session.
- Mark the service for deletion because the matching UDDI service was not found in the UDDI registry. Customize the resource appropriately and then execute the customization file. Later, use AquaLogic Service Bus Console to detach the service from the UDDI registry.

  The disadvantage of this approach is that besides UDDI, the production environment values must also be present in the customization file.

  After detaching the service from the registry, you can download the UDDI-related information from the UDDI registry and then enable automatic synchronization, if required.

Importing an existing service into a production system

In this scenario, do one of the following:

- Overlay the production UDDI service during import when a UDDI registry (or identical UDDI registries - in terms of service key) is used. You can then force a UDDI re-synchronization of the service in the current session with an API call.
- Use the import script to skip importing this service to the production system. This is the only option if the UDDI production keys are different.

  However, when you skip importing the service, non-UDDI sourced data in the service may be wrong. So, you must directly update such information in the production service definition instead of importing the latest service.