BEA AquaLogic® Service Bus

Interoperability With EJB Transport

Version 3.0
Revised: February 2008

Contents

EJB Transport

Using the EJB transport, ALSB supports native RMI invocation of stateless session beans deployed on WebLogic Server 8.1, 9.0, 9.1, or 9.2. It allows transactional and secure communications. You can also leverage the EJB transport to expose an EJB as a Web Service through ALSB.

This section includes the following topics:

Introduction

In ALSB, you can configure business services to use the EJB transport. The EJB transport is fully integrated into the ALSB configuration, management, monitoring, and test consoles. You can use business services configured with the EJB transport for Publish, Service Callout, and service invocations. You cannot create proxy services that use the EJB transport.

The EJB transport provides the following capabilities:

Transactional Integrity
: You can call an EJB business service in the context of a global transaction. The EJB transport can also suspend or start a global transaction before invoking an EJB.

Security Propagation
: The security context established at the beginning of a message flow, from an ALSB client is propagated to the other system. In other words, an incoming SOAP over HTTP request to ALSB that requires authentication is authenticated by ALSB and the authenticated subject can then be propagated to the EJB server.

HTTP Tunneling and Encrypted Communication
: You can access EJBs that are behind a fire wall with HTTP tunneling. For additional security, you can use SSL to encrypt all of the communications with the EJB Server.

JNDI Provider
: EJB transport leverages the JNDI provider—an ALSB resource. The JNDI provider defines communication protocols and security credentials for accessing remote servers. A JNDI provider can be reused by multiple EJB business services. This provides a centralized way for administrators to manage remote EJB server configurations.

For information about JNDI provider resources, see Global Resources in *Using the AquaLogic Service Bus Console*.

High Performance Caching
: The EJB transport is built on high performance cache. This allows the reuse of established connections and minimizes EJB stubs lookups.

Failover and Load Balancing
: The EJB transport can take advantage of scenarios in which the same EJB is deployed in multiple domains or on a cluster for load balancing or failover or both.

Advanced XML to Java Binding Capabilities
: The EJB transport leverages the WebLogic Server JAX-RPC stack to perform Java to XML bindings. The JAX-RPC stack is a high performance engine that supports advanced Java objects such as XML Beans. If the Java type is not recognized by the stack, an extension mechanism is provided to facilitate support of these Java types. For information about this extension mechanism (using the converter classes), see Supported Types and Converter Class.

Intelligent Retries
: The EJB transport makes retry decisions based on the nature of the failure that can occur during the invocation of an EJB.

Invoking EJBs from ALSB

Before you can configure a business service in ALSB, you must register a JNDI provider resource and a client JAR resource. This section describes how to design and configure an EJB transport business service in ALSB.

Register a JNDI Provider Resource

A JNDI Provider resource allows you to specify the communication protocols and security credentials used to retrieve EJB stubs bound in the JNDI tree of remote WebLogic 8.1 or 9.x domains. (For more information how to setup a JNDI tree, see Programming WebLogic JNDI in the BEA WebLogic Server documentation.)

Typically, the target EJB is not located in the same domain as ALSB. In this case, you must register a JNDI Provider resource. When the EJB is located in the same domain, you can define a provider to specify credentials and take advantage of stubs caching, although it is optional in this case.

The JNDI provider has a high performance caching mechanism for remote connections and EJB stubs. The preferred communication protocol from ALSB to a WebLogic Server domain is `t3` or `t3s`. If messages need to go through a fire wall, you can use HTTP tunneling. For more information about HTTP tunneling, see HTTP Tunneling and Encrypted Communication.

Notes:

- Although it is possible to use a WebLogic Server foreign JNDI Provider, BEA recommends that you do not.
- 2-way SSL is not supported.
- EJB transport provider does not support CLIENT CERT to look-up JNDI tree or access a method on an EJB.

Adding a JNDI Provider

For information about registering and configuring a JNDI provider resource in ALSB, see “Adding JNDI Providers” see Global Resources in *Using the AquaLogic Service Bus Console*.

Register an EJB Client JAR Resource

A client JAR must be registered as a resource in ALSB. It is therefore part of the ALSB configuration and can be exported from and imported into a project.

An EJB client JAR file must contain the interfaces and classes needed by ALSB to access an EJB. This includes the remote and home interfaces and any dependent types to which the client is exposed, such as method parameter types or application exceptions.

If your business service requires converter classes, you can register a JAR file containing the converter classes as an ALSB resource and subsequently use these classes to help map parameter and return value types to Java classes that can be mapped to XML. Alternatively, you can package these converter classes in the EJB client JAR. For information about converter classes, see Converter Classes.

Consider the following guidelines when using EJB client JARs:

- Adding Home and remote interfaces in the system classpath is bad practice and is not supported by ALSB.
- BEA recommends that you keep the client JAR size small, include a single home interface per JAR and not register the entire ejb-jar file.
- You can use WebLogic Workshop to obtain a client JAR for EJBs deployed on WebLogic Server 8.1 or 9.x.
- Client-jars compiled with JDK 1.4 or later are supported.

Adding a Client or Converter JAR

For information about registering and configuring a JAR resource in ALSB, see “Adding a JAR” in JARs in *Using the AquaLogic Service Bus Console.*

Create a Service Account (Optional)

If the EJB methods are protected, you can specify the credentials you want to use for the invocations. Those credentials are often different than the credentials used by the JNDI provider. For information about adding and using service accounts, see Service Accounts in *Using the AquaLogic Service Bus Console*.

Locate an EJB in the JNDI Tree

If you do not know the JNDI name for an EJB, you can browse the EJB Server JNDI tree. For information about browsing the JNDI tree using the WebLogic Server Administration Console, see:

- JNDI in the *WebLogic Server 8.1 Administration Console Online Help* (for WebLogic Server 8.1)

- View objects in the JNDI tree in the *WebLogic Server 9.2 Administration Console Online Help* (for WebLogic Server 9.x)

Create an EJB Business Service

This section provides information about creating a business service that uses the EJB transport.

General Configuration

1. Open the ALSB Console and in an active session, select Project Explorer from the left navigation panel. The **Project View** page is displayed.

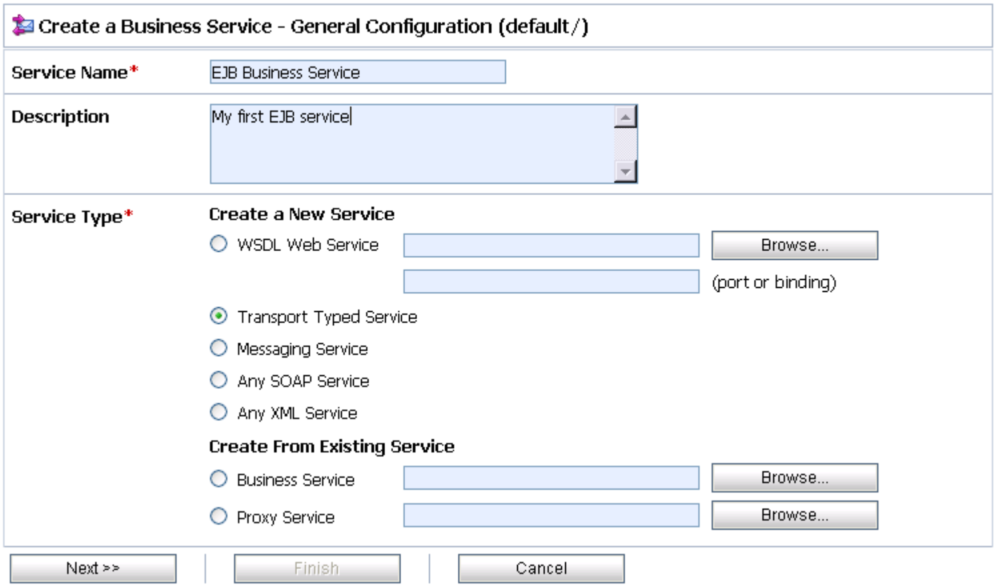
2. Select the project in which you want to create the business service. A page in which you can create a business service is displayed—create a new business service.

3. On the General Configuration page, as shown in the following figure, enter a name for the business service and select the **Transport Typed Service** as the **Service Type**.

   An EJB business service is a *Transport Typed Service*, meaning that the type of the transport is determined by the configuration of the service. The EJB transport is currently the only such transport type. You can add other transports by using the ALSB Transport SDK.

   An entry in the Description field is optional.

Figure 1-1 Create a Business Service - General Configuration

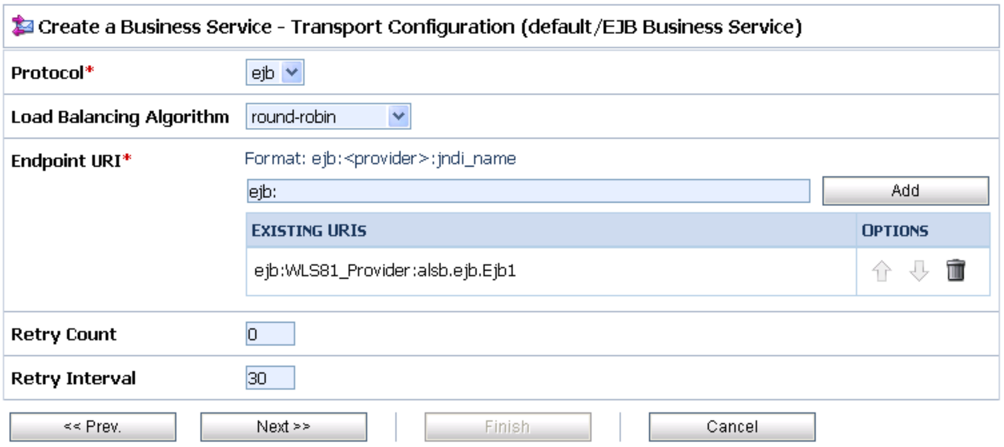


Create a Business Service - General Configuration (default/)

Service Name*	EJB Business Service
Description	My first EJB service
Service Type*	Create a New Service
	( ) WSDL Web Service
	(port or binding)
	(●) Transport Typed Service
	( ) Messaging Service
	( ) Any SOAP Service
	( ) Any XML Service
	Create From Existing Service
	( ) Business Service
	( ) Proxy Service

4. Click Next to open the transport-specific configuration page.
5. In the Transport Configuration page, select ejb as the Protocol.

Figure 1-2 Create a Business Service—Transport Configuration



Create a Business Service - Transport Configuration (default/EJB Business Service)

Protocol*	ejb
Load Balancing Algorithm	round-robin
Endpoint URI*	Format: ejb:<provider>:jndi_name
	ejb:
Retry Count	0
Retry Interval	30

EXISTING URIS	OPTIONS
ejb:WLS81_Provider:alsb.ejb.Ejb1	

6. Enter the Endpoint URI and add it to the list of EXISTING URIs. To build the URI you need the name of the provider you used in Adding a JNDI Provider and the location of the EJB home interface in the JNDI tree you determined in Locate an EJB in the JNDI Tree:

   `ejb:`*`provider`*`:`*`jndi_name`*

   If the EJB is deployed locally, you need not provide a JNDI provider name. In this case, the URI format is:

   `ejb::`*`jndi_name`*

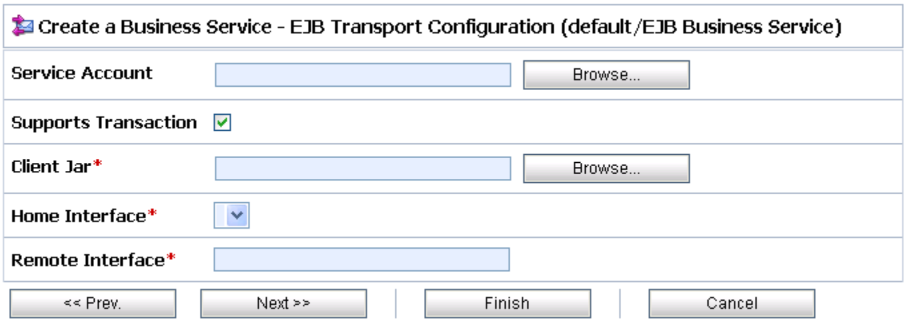
7. As for any ALSB transport, you can also specify the Load Balancing Algorithm, Retry count, Retry Interval and specify multiple URIs for failover. See Retries and Failover.

8. Click Next to open the EJB transport-specific configuration page.

EJB Transport-Specific Configuration

After completing the general configuration of the business service you specify EJB transport-specific information such as the Home and Remote interfaces. The EJB Transport Configuration page in the ALSB Console is shown in the following figure.

Figure 1-3 Create a Business Service— EJB Transport Configuration



Create a Business Service - EJB Transport Configuration (default/EJB Business Service)

Service Account		Browse...
Supports Transaction	✓	
Client Jar*		Browse...
Home Interface*		
Remote Interface*		

To Configure the EJB Transport

1. Optionally select a Service Account.

   If the EJB methods are protected and you defined a service account as described in Create a Service Account (Optional), click Browse to locate the appropriate Service Account.

2. By default, the Supports Transaction option is selected. This specifies that the EJB supports transaction. If you do not want to propagate transactions, or if the EJB does not support transactions, deselect Supports Transaction.

For information about transaction processing with the EJB Transport, see Transaction Processing, Retries, and Errors Handling.

3. Select the Client JAR—browse and select the Client JAR you registered previously, as described in Adding a Client or Converter JAR.

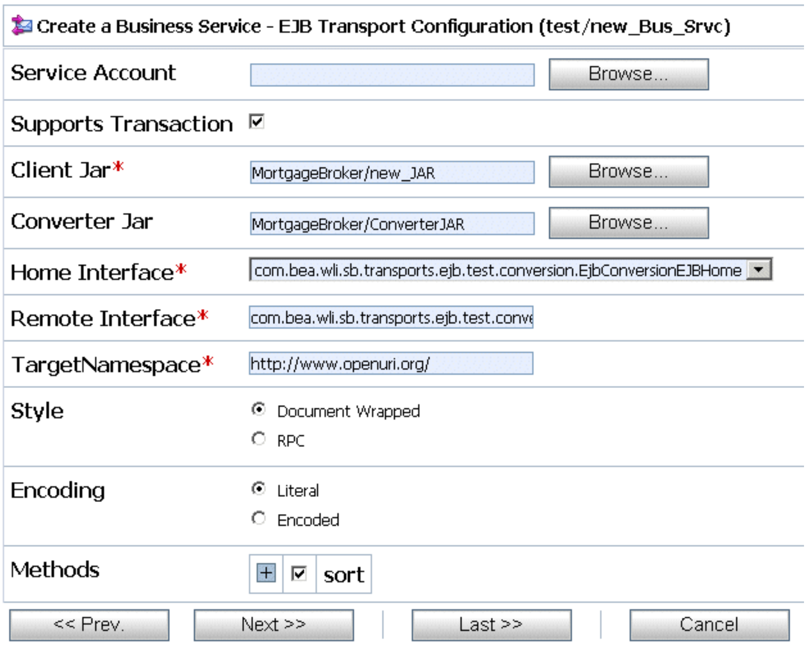
   When you select a Client JAR, a list of its Home Interface is displayed on this page. Additionally, a Converter JAR field is displayed.

4. If required, select the Converter JAR—browse and select the Converter JAR you registered previously, as described in Adding a Client or Converter JAR.

5. Select the Home Interface from the list of interfaces provided in the Home Interface field.

   Notice that the Remote Interface is automatically deduced from the Home Interface and the configuration page is refreshed. The Remote Interface field is populated and other options are provided that allow you to control the interface of the service and the WSDL generated when you finish configuration of this business service.

Figure 1-4 Create a Business Service— EJB Transport Configuration after Selecting the Home Interface



Create a Business Service - EJB Transport Configuration (test/new_Bus_Srvc)

Service Account	
Supports Transaction	✓
Client Jar*	MortgageBroker/new_JAR
Converter Jar	MortgageBroker/ConverterJAR
Home Interface*	com.bea.wli.sb.transports.ejb.test.conversion.EjbConversionEJBHome
Remote Interface*	com.bea.wli.sb.transports.ejb.test.conve
TargetNamespace*	http://www.openuri.org/
Style	[✓ Document Wrapped] RPC
Encoding	[✓ Literal] Encoded
Methods	✓ sort

EJB Business Service Interface Configuration

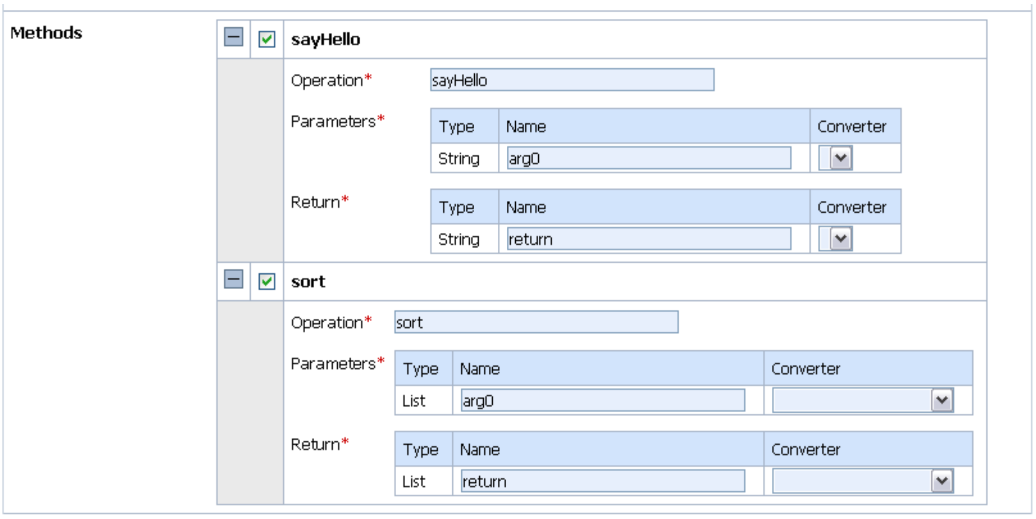
An EJB business service is a Transport Typed Service, which means the type of the transport is determined by the configuration of the service.

The type of an EJB business service is equivalent to a SOAP XML service—in other words, you can use an EJB business service like any other SOAP XML business service. A WSDL is generated when you save the EJB Transport Configuration.

The WSDL is generated based on the interface of the EJB. The EJB transport configuration page provides configuration options for you to control the interface of the service and the WSDL that is generated. To do so, complete the configuration on the EJB Transport Configuration page as shown in the preceding figure:

1. TargetNamespace—Specify the target namespace of the WSDL.
2. Style—You can select Document Wrapped or RPC.
3. Encoding—Select Literal or Encoded.
4. The methods displayed are those of the EJB Remote Interface you selected. For example, the following figure displays two methods: `sayHello` and `sort`.

Figure 1-5 Create a Business Service— EJB Transport Configuration, Expanded Methods Configuration



Methods

sayHello ✓

Operation* sayHello

Parameters*

Type	Name	Converter
String	arg0	

Return*

Type	Name	Converter
String	return	

sort ✓

Operation* sort

Parameters*

Type	Name	Converter
List	arg0	

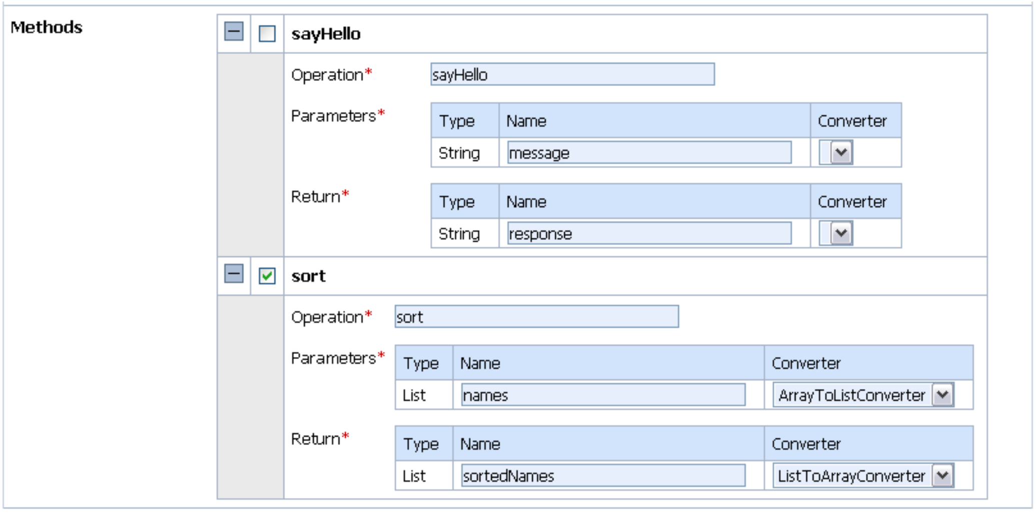
Return*

Type	Name	Converter
List	return	

5. You can exclude the methods you do not want to expose by unchecking the check box associated with the method names.
6. You can change the default operation name for a given method. (By default, the operation name is the method name.) If an EJB contains methods with same name, you must change the operation names so that they are unique—WSDLs require unique operation names.
7. You must exclude the methods with parameters or return types that are not supported by the JAX-RPC stack or you must associate such arguments with Converter Classes.

   The following figure shows an example of a custom methods configurations.

Figure 1-6 Create a Business Service— EJB Transport Configuration, Customized Methods Configuration



Methods

☐ sayHello

Operation*: sayHello

Parameters*

Type	Name	Converter
String	message	

Return*

Type	Name	Converter
String	response	

✓ sort

Operation*: sort

Parameters*

Type	Name	Converter
List	names	ArrayToListConverter

Return*

Type	Name	Converter
List	sortedNames	ListToArrayConverter

8. Click Next. Save the service and activate the session.

Note: If the credentials or transaction settings are different between the methods for a given EJB, you can leverage the ability to customize the methods for a given business service, and create a business service per method. This gives you fine-grained control over transactions and credentials.

Invoking EJB Business Services

An EJB business service can be used as a SOAP XML business service. You can publish to, route to, or callout to an EJB business service. If you need transaction support, set the QoS to *Exactly-Once*. See Transaction Processing, Retries, and Errors Handling.

You can also use the test console to validate your configuration and to help you to determine the shape of the XML request.

Exposing EJBs as Web Services

You can leverage the EJB transport to easily expose EJBs as Web Services.

Note: You cannot create a proxy service from an existing EJB business service—you must first get the WSDL generated from the EJB business service, and then create the proxy service based on that WSDL. To do so, complete the following steps:

1. Create an EJB business service pointing to the EJB you want to expose, as described in Create an EJB Business Service.
2. From the service details page on the ALSB Console, get the WSDL for the EJB business service.

   The WSDL is contained in a JAR file. You can obtain the WSDL only if there is no pending session.
3. Extract the WSDL from the JAR and register it as a WSDL resource. For information about creating WSDL resources, see WSDLs in *Using the AquaLogic Service Bus Console*.

   If the configuration of the business service changes, a new WSDL is generated. If that happens, you must get the new WSDL and re-register it as a WSDL resource.
4. Create a SOAP XML proxy service based on the WSDL.
5. Edit the proxy service pipeline and route to the EJB business service.

You can now invoke the EJB as a Web Service with no need for purchasing an expensive Web Service toolkit or carrying out intrusive actions on the EJB server.

Advanced Topics

This section includes information about EJB transport that will help you understand how EJB business services behave at run time depending on how they are configured at design time.

Transaction Processing, Retries, and Errors Handling

Transactions

The EJB transport can create, suspend, and propagate transactions. The transaction between ALSB and the EJB server are XA transactions. If you use transactions with HTTP tunneling or have a dedicated communication channel and the EJBs are deployed on 8.1 servers, you must set the *security interoperability* mode for the transaction manager to `performance`. For information about setting the security interoperability mode and other transaction configurations, see Configuring Transactions in *Programming WebLogic JTA*.

For the deployment descriptors to be set appropriately for XA capable resources (JMS, TUXEDO, EJB), you must set the XA attribute on the referenced connection factory before creating a proxy service.

To determine the behavior of the EJB business service, considerations include whether the proxy service pipeline has a transactional context, and what quality of service (QoS) settings are specified in the pipeline when invoking the service:

QoS Best-Effort

If *Best Effort* QoS is specified in the pipeline, no transaction is propagated to the EJB—any ongoing transaction is suspended before invocation, and resumed after invocation.

QoS Exactly-Once

If *Exactly Once* QoS is specified in the pipeline, and

If the EJB does not support transactions (that is, if the Supports Transaction option on the EJB transport configuration page is unchecked), no transaction is propagated to the EJB. As in the case of *Best Effort*, any ongoing transaction is suspended before invocation and resumed afterwards.

or

If the EJB supports transactions (that is, if the Supports Transaction option on the EJB transport configuration page is checked), the EJB is invoked in the context of a transaction—any ongoing transaction is propagated to the EJB. If no transaction is present, a transaction is created before invocation and committed afterwards.

For more information about QoS in ALSB services, see “Quality of Service” in Modeling Message Flow in *AquaLogic Service Bus User Guide*.

Retries and Failover

Assuming that the EJB business service is configured for retries or failovers, the EJB transport distinguishes the following types of exceptions:

- Runtime Exceptions or Remote Exceptions—typically unexpected fatal errors or communication exceptions
- Exception raised by the JAX-RPC engine—exceptions that occur during the XML to Java conversion
- EJB Checked Exceptions—exceptions declared in the EJB method signature specific to the EJB implementation; also called Business Exceptions

Retries and failover are based on the type of errors and also in the QoS:

QoS Best-Effort

If a run-time or remote exception is thrown, the EJB transport attempts retries or failovers.

If an exception occurs in the JAX-RPC engine, an error is raised to the pipeline and no retries or failover attempts are made.

If an EJB Checked Exception is thrown, an error is raised to the pipeline and no retries or failover attempts are made.

QoS Exactly-Once

If a run-time or remote exception is thrown and the ongoing transaction has been set as *rollback only* (likely by the EJB container), it means the EJB container has been reached and a fatal error either occurred within the EJB container or the EJB. In this case, no retries or failover attempts are made and an error is raised to the pipeline.

If a runtime or remote exception is thrown but the ongoing transaction has not been set as *rollback only*, it means an error occurred before the invocation of the EJB container and the EJB transport will attempt retries or failovers. Note that in this case, the EJB transport still respects the e*xactly-once* semantic.

If an exception occurs in the JAX-RPC engine, the EJB transport sets the ongoing transaction to *rollback only* and an error is raised to the pipeline; no retries or failover attempts are made.

If an EJB Checked Exception is thrown, an error is raised to the pipeline and no retries or failover attempts are made.

See “Transactions” on page 1-12 for other repercussions of QoS specifications for an EJB business service.

Error Handling

When throwing a checked exception, according to the EJB specifications, the ongoing transaction can be specified as *rollback only*.

If the ongoing transaction is set as *rollback only* by the EJB developer, the transaction is eventually rolled back by its creator (most likely the proxy service).

If the ongoing transaction is not set to *rollback only*, and a checked exception is raised, it is important to catch EJB checked exceptions in the pipeline with an error handler. If those exceptions are not caught, the pipeline errors are propagated back to the proxy service. The proxy service, in turn, is likely to rollback the ongoing transaction (depending of the transport implementation)—this may not be the intended result.

For example, assume you have an EJB with the following method:

```
public void withdrawFunds(float amount) throws RemoteException,
InsufficientFundsException {…}
```

Also assume that when an `InsufficientFundsException` exception is thrown, the EJB does not set the current transaction as *rollback only*. In most scenarios, it is wrong to allow the proxy service to roll back the transaction—you may need to configure an error handler in the pipeline to catch the error and avoid this scenario.

Supported Types and Converter Class

The EJB transport is responsible for the XML←→Java conversion. The conversion is performed by the WebLogic Server JAX-RPC engine.

The EJB transport natively supports the following types:

- Primitive types
- XmlObject (both Apache and BEA versions)
- Schema generated XMLBeans (both Apache and BEA versions)
- JavaBean classes

For the full list of natively supported types, see Data Types and Data Binding in *Programming Web Services for WebLogic Server*.

An EJB method can use parameters/return types that are either not supported by the JAX-RPC engine (an error is reported at design time), or that do not map directly to XML (errors occur at run time). The most commonly used unsupported types are:

- "Object", "Object[]"
- Java Collections as they are not strongly-typed (for example, List, Set)
- Java classes that do not follow the JavaBean pattern (for example, Map)

You can write a custom converter class than converts those types into types more suitable for XML←→Java conversions. The EJB transport supports custom converter classes.

Converter Classes

A Converter class is a Java class that implements and conforms to the contract defined by the `com.bea.wli.sb.transports.ejb.ITypeConverter` Java interface of the ALSB public API. For information about the `ITypeConverter` Java interface and other ALSB APIs, see the ALSB Javadoc.

To use a converter class for an EJB business service, you must:

1. Create a converter class by implementing and compiling the interface.
2. Add the converter class to the client JAR or to a converter class JAR file (See "Adding a Client or Converter JAR" on page 1-4).
3. When customizing the method configuration during the creation of an EJB business service, navigate to one of the parameter/return types and select the desired converter. See step 7 in "EJB Business Service Interface Configuration" on page 1-9—the ALSB Console displays a list of the converters available that can be applied to a particular parameter/return type.

Troubleshooting

The information in this section is provided to help you troubleshoot problems when designing or running an EJB business service.

Enabling Debug Mode

The EJB transport uses the same logger as other ALSB transports. To enable the debug mode, before starting the server, edit the `wlidebug.xml` file in the domain directory and set the category wli-sb-transports-debug to `true`. For more information about the `wlidebug.xml` file and the debug flags, see Debugging AquaLogic Service Bus in *AquaLogic Service Bus User Guide*.

Temp Directories

During design time, the EJB transport generates files in the subfolder alsbejbtransport and subfolders prefixed with appcgen_ in the `temp` directory. It is safe to delete those folders and files, and sometimes may be useful to check them to determine what went wrong during activation.

Deployed Application

When an EJB business service is created an application is deployed on the AquaLogic Server. You can use the WebLogic Server Administration Console to monitor and tune this application. The name of EJB business service applications is prepended with `ALSB EJB`, which is followed by the WSDL type and an auto generated suffix.

Errors

The following items may help in the event that you need to troubleshoot a problem with an EJB business service:

- The following error when creating a business service is due to a Windows operating system limitation—paths containing more than 255 characters are not supported:

  ```
  The system cannot find the path specified):Probably the string length of
  the path of the file being extracted was too long
  ```

  You can try to reduce the path length by creating a shorter path to the ALSB domain, or you can use the following option to override the WebLogic Server `temp` directory when starting the server:

  ```
  -Dweblogic.j2ee.application.tmpDir=$desired_short_dir
  ```

- If you get an XML marshalling error when invoking an EJB business service and you believe the request to be valid against the service WSDL, you probably need to write a converter class. For information, see "Converter Classes" on page 1-15.

- If the EJB interfaces and stubs are changed on the remote server, the first time you try to invoke the new EJB, an error is thrown. Those changes on the remote server are not visible to ALSB—it tries to invoke the cached EJB stubs, which are no longer valid. However, when the invocation error occurs, the transport assumes that those stubs are now invalid, and remove them from the cache—in this way, the error is prevented on subsequent attempts to invoke the EJB. To avoid this first-time error, you can reset the JNDI Provider in the ALSB Console.

- For HTTP tunneling between WebLogic Server 9.2 and WebLogic Server 8.1 to work, you must set the `t3-server-abbrev-table-size` element to `255` in the `config.xml` file in the ALSB domain, as shown in the following code snippet:

```
<server>
    <name>AdminServer</name>
    <ssl>
      <name>AdminServer</name>
      <enabled>true</enabled>
    </ssl>
        <t3-server-abbrev-table-size>255</t3-server-abbrev-table-size>
    <listen-address></listen-address>
  </server>
```