# BEA AquaLogic® Service Bus

## Interoperability Solutions for JMS

Version: 2.6 RP 1
Document Revised: November 2007

# Contents

CHAPTER 1

# Interoperability with JMS

The following sections describe features and concepts related to interoperability between ALSB and WebLogic JMS:

- Overview of JMS Interoperability
- Asynchronous Request-Response Messaging
- Using SOAP-JMS Transport
- Naming Guidelines for Domains, WebLogic, and JMS Servers
- Specifying the JMS Type for Services
- WSDL-Defined SOAP Fault Messages

## Overview of JMS Interoperability

Java API for XML-Remote Procedure Call (JAX-RPC) is considered the core Java API to build and deploy Web services using J2EE. JAX-RPC provides a simple, robust platform for building Web services applications by abstracting the complexity of mapping between XML types and Java types and the lower-level details of handling XML SOAP messages from the developer. JAX-RPC introduces a method call paradigm by providing two programming models:

- A server-side model for developing Web services endpoints using Java classes or stateless EJB components
- A client-side model for building Java clients that access Web services as local objects.

JAX-RPC mandates the use of SOAP and interoperability with other Web services built with other technologies. If you already have a stateless session EJB or a Java class that performs your business logic, J2EE 1.4 lets you expose it as a service in a standard manner using JAX-RPC.

AquaLogic Service Bus is certified to work with the following JMS implementations:

- WebLogic Server 9.x JMS
- IBM WebSphere MQ/JMS 5.3
- TIBCO Enterprise Message Service™ 4.2

All ALSB service types support JMS transport. Proxy services and business services must be configured to use JMS transport as described in the Proxy Services and Business Services sections of *Using the AquaLogic Service Bus Console*.

For information about ALSB service types and the transports for each of the service types, see "Selecting a Service Type" in Modeling Message Flow in AquaLogic Service Bus in the *BEA AquaLogic Service Bus User Guide*.

For information about WebLogic Server 9.x JMS, see:

- Managing Your Applications in *Programming WebLogic JMS*
- Configure JMS Servers in the *WebLogic Server Administration Console Online Help*

**Note:** ALSB version 2.6 now supports the MQ Extended Transactional Client which is vital for remote transactional support configuration.

# Asynchronous Request-Response Messaging

Messaging can be categorized as follows:

- One-way
- Synchronous request-response
- Asynchronous request-response

However, messaging over JMS is only one-way or via asynchronous request-response. Asynchronous request-response messaging using JMS is an alternative to messaging using HTTP or HTTP(s).

Using asynchronous request-response messaging has the following advantages:

- The request thread does not get blocked while waiting for the response. This removes a thread management issue that can occur when numerous blocking request-response invocations are made. However, HTTP and HTTP(s) support a non blocking mode of operation.
- The messaging is more reliable than HTTP because it can be:
  - Persisted on disk
  - Queued when the service is not available
  - Re-delivered if the server has an error or fails when the message is being processed

If you are using IBM WebSphere MQ, asynchronous request-response messages may be the best approach for interacting with some mainframes. The asynchronous service must echo the correlation ID or the message ID depending on the JMS request-response pattern that you use. The format of either ID used by ALSB is compatible with IBM WebSphere MQ and with target services that use MQ native interfaces. For more information, see Chapter 2, "Understanding Message ID and Correlation ID Patterns for JMS Request/Response".

Asynchronous request-response messages are handled by the outbound and inbound transports. That is, the message flow, except for the `$outbound` and `$inbound` transport specific data, does not distinguish between JMS request-response and HTTP request-response.

AquaLogic Service Bus supports bridging between synchronous and asynchronous request and response. For example, a proxy service can be invoked using HTTP, and the proxy service routes to a JMS request-response business service. This is called synchronous-to-asynchronous service switching.

# Using SOAP-JMS Transport

You can use JMS for SOAP transport instead of HTTP, as SOAP is transport-agnostic. SOAP JMS transport is necessary especially for enterprise customers. AquaLogic Service Bus supports SOAP messages with JMS request-response. AquaLogic Service Bus supports interoperation with WebLogic Server SOAP-based clients and services.

JMS is also an approach for reliable messaging.

This section contains examples for interoperating with SOAP-JMS as follows:

- Interoperating with BEA WebLogic Server 9.x
- Interoperating with BEA WebLogic Workshop 8.1
- Interoperating with BEA WebLogic Server 8.1

# Interoperating with BEA WebLogic Server 9.x

When you use JMS binding to configure a service in WebLogic Server 9.x, use the following SOAP-JMS URL format with WebLogic Server:

```
jms://host:port/contextURI/serviceName?URI=destJndiName
```

When you configure the service in AquaLogic Service Bus, the URL must have the following format:

```
jms://host:port/factoryJndiName/destJndiName
```

Both formats use the same `destJndiName`. The `factoryJndiName` must be the JNDI name of an existing QueueConnectionFactory in the target WebLogic Server.

When you invoke BEA WebLogic Server services from AquaLogic Service Bus, you must set the URI as a JMS property with the value as `/contextURI/serviceName`, inside the message flow on the outbound variable (`$outbound`) before a request is sent to the business service. Use the Transport Headers action to set this property.

For information about setting `$outbound`, see the "Inbound and Outbound Variables" section in Message Context of the *AquaLogic Service Bus User Guide*.

When a WebLogic Server 9.x Web Services client invokes an AquaLogic Service Bus proxy service, the URI property is ignored. However, it can be passed through to an invoked service using the pass through options of the Transport Headers action.

AquaLogic Service Bus can only invoke WebLogic request-response services running on version 9.2 or later. However, it can also invoke one-way JMS services.

# Interoperating with BEA WebLogic Workshop 8.1

When you use the JMS binding to configure a business service in BEA WebLogic Workshop 8.1, use the following the SOAP/JMS URL format with BEA WebLogic Workshop:

```
jms://host:port/factoryJndiName/destJndiName?URI=/process/myprocess.jpd
```

AquaLogic Service Bus always uses JMS URLs with the following format:

```
jms://host:port/factoryJndiName/destJndiName
```

When you invoke BEA WebLogic Workshop services from AquaLogic Service Bus, you must set the URI as a JMS property inside the message flow on the outbound variable (`$outbound`) before it is sent. Use the Transport Headers action to set this property.

When a WebLogic Workshop client (for example, a control) invokes an AquaLogic Service Bus proxy service, the URI property is ignored. However, it can be passed through to an invoked service using the pass through options of the Transport Headers action.

AquaLogic Service Bus cannot invoke WebLogic 8.1 JMS request-response services. However, it can invoke one-way JMS services.

For information about setting `$outbound`, see "Inbound and Outbound Variables" in Message Context, in the *AquaLogic Service Bus User Guide*.

## Interoperating with BEA WebLogic Server 8.1

When you use the JMS binding to configure a business service in WebLogic Server 8.1, use the following SOAP-JMS URL format with WebLogic Server:

```
jms://host:port/factoryJndiName/destJndiName?URI=/contextURI/serviceName
```

This format differs from the AquaLogic Service Bus JMS URL format shown in the preceding sections. When invoking BEA WebLogic Server services from AquaLogic Service Bus, you must set the URI as a JMS property inside the message flow on the outbound variable (`$outbound`) before a request is sent to the business service. You can use the Transport Headers action to set this property.

When a WebLogic Server 8.1 Web Services client invokes a AquaLogic Service Bus proxy service, the URI property is ignored. However, it can be passed through to an invoked service using the pass through options of the Transport Headers action.

For information about setting `$outbound`, see "Inbound and Outbound Variables" in Message Context, in the *AquaLogic Service Bus User Guide*.

# Naming Guidelines for Domains, WebLogic, and JMS Servers

If you are working with more than one domain is involved, make sure your configuration conforms to the following requirements:

- WebLogic Server instances and domain names are unique.
- WebLogic JMS server names are uniquely named across domains.
- If a JMS file store is being used for persistent messages, the JMS file store name must be unique across domains.

Note the following rules for JMS Server names:

- You cannot have duplicate JMS server names within the same domain. If you do, when messages are sent to a destination at a particular JMS server, ALSB cannot determine to which server the message should be sent.
- If you are using Store and Forward (SAF), duplicate JMS Server names in different domains do not pose a problem.
- In the case of cross-domain communication, duplicate JMS server names can be a problem when you use the `ReplyTo` function. The `ReplyTo` message sent from a given domain is returned to the JMS server on the same domain that received the message instead of being returned to the domain that sent the original message.

For more information on how to configure and manage WebLogic JMS:

- Managing Your Applications in *Programming WebLogic JMS*
- Configure JMS Servers in the *WebLogic Server Administration Console Online Help*

For information about WebLogic Server domains, see *Understanding Domain Configuration*.

# Specifying the JMS Type for Services

To support interoperability with heterogeneous endpoints, AquaLogic Service Bus allows you to control the content type used, the JMS type used, and the encoding used when configuring message flows. The JMS type can be byte or text. For more information, see "Content Types, JMS Type, and Encoding" in Modeling Message Flow in AquaLogic Service Bus in the *AquaLogic Service Bus User Guide*.

# WSDL-Defined SOAP Fault Messages

When consuming a WSDL that explicitly defines a fault, the WebLogic clientgen tool generates a subclass of `java.lang.Exception` for the XML fault type. When the WebLogic Server JAX-RPC stack inspects a SOAP response message and determines that the response message contains a SOAP fault, it tries to map the fault to a clientgen-generated exception Java class.

For example, if a WSDL contains the definitions shown in the following listing, the clientgen tool generates a Java class `com.bea.test.TheFaultType` that extends `java.lang.Exception`. A JAX-RPC client can catch `com.bea.test.TheFaultType` when invoking the related method of the service stub.

**Listing 1-1 Sample WSDL Definitions**

```
<definitions ... xmlns:s0="http://www.bea.com/test/">
   ...
   <types>
      <xsd:schema targetNamespace="http://www.bea.com/test/">
         ...
         <xsd:complexType name="theFaultType">
            <xsd:sequence>
               <xsd:element name="ID" type="xsd:int" />
               <xsd:element name="message" type="xsd:string" />
            </xsd:sequence>
         </xsd:complexType>
         <xsd:element name="theFault" type="theFaultType" />
      </xsd:schema>
   </types>
   ...
   <message name="theFaultMessage">
      <part element="s0:theFaultPart" name="theFault" />
   </message>
   ...
   <binding ...>
      <operation ...>
         <soap:operation soapAction="..." style="document" />
         <input ...>
            ...
         </input>
         <output ...>
            ...
         </output>
         <fault ...>
            <soap:fault name="theFaultPart" use="literal" />
         </fault>
      </operation>
   </binding>
   ...
</definitions>
```

The SOAP message must contain a fault of the correct format so that the JAX-RPC stack throws the correct exception. If the fault is constructed from inside an ALSB message flow, you must:

1. Replace the node for the `$body` variable with the following sample SOAP:

**Listing 1-2 Sample SOAP**

```
<soap-env:Body>
      <soap-env:Fault>
         <faultcode
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">soap:Server</faultc
ode>
         <faultstring>Some literal string</faultstring>
         <detail>
            <test:theFault>
               <test:ID>Any user defined code</test:Id>
               <test:message>A specific literal message</test:message>
            </test:theFault>
         </detail>
      </soap-env:Fault>
</soap-env:Body>
```

   where:

   – `soap-env` is the system prefix for the namespace `http://schemas.xmlsoap.org/soap/envelope/`

   – `test` is the prefix for the namespace `http://www.bea.com/test/`

     If the prefix `test` is not already known to ALSB, you must declare it.

2. Configure a reply action with failure.

For information about configuring Reply Actions in the AquaLogic Service Bus Console, see Proxy Services Actions in *Using the AquaLogic Service Bus Console*.

The clientgen tool is used to generate the client-side artifacts, such as the JAX-RPC stubs, needed to invoke a Web Service. See Ant Task Reference in *Programming Web Services for WebLogic Server*.

CHAPTER 2

# Understanding Message ID and Correlation ID Patterns for JMS Request/Response

JMS is a standard API for accessing enterprise messaging systems. WebLogic JMS:

- Enables Java applications sharing a messaging system to exchange messages.
- Simplifies application development by providing a standard interface for creating, sending, and receiving messages.

For an overview and features of JMS, see JMS Interoperability and Configuring and Managing WebLogic JMS.

This chapter describes the Message ID and Correlation ID patterns supported in AquaLogic Service Bus for JMS request-response and describe how AquaLogic Service Bus uses these patterns to interoperate with Java API for Remote Procedure Call (JAX-RPC) Web Services. Examples and known limitations are also provided.

The following sections are included in this chapter:

# Overview of JMS Request-Response and Design Patterns

Messaging provides high-speed, asynchronous, program-to-program communication with guaranteed delivery and is often implemented as a layer of software called Message Oriented Middleware (MOM).

In enterprise computing, messaging makes communication between processes reliable, even when the processes and the connection between them are not so reliable. Processes may need to communicate for the following reasons:

- One process has data that needs to be transmitted to another process.
- One process needs to remotely invoke a procedure in another process.

Asynchronous request-response messages are the best approach to interacting with some mainframes, if you are using IBM WebSphere MQ.

## Patterns for Messaging

Messaging patterns describe the format of messages that flow between parts of a system built with a MOM. There are several types of messages as follows:

- A Command Message enables procedure call semantics to be executed in a messaging system.
- A Document Message enables a messaging system to transport information, such as the information that should be returned to a sender as a result of a command message.
- An Event Message uses messaging to perform event notification.
- A Reply Message handles the semantics of remote procedure call results, that require the ability to handle both successful and unsuccessful outcomes.
- A Reply Specifier enables a program making a request to identify the channel through which a reply should be sent.
- A Correlation Identifier enables a requesting program to associate a specific response with its request. When the data to be conveyed spans several messages, a Sequence Identifier enables the receiver to accurately reconstruct the original data.
- Message Expiration enables a sender to set a deadline by which the message should either be delivered or ignored.
- Message Throttle enables a receiver to control the rate at which it receives messages.

In the case of AquaLogic Service Bus, each reply message should contain a unique identifier called the correlation ID, that correlates the request message and its reply.

When the caller creates a request message, it assigns a unique identifier to the request that is different from those for all other currently outstanding requests (for example, requests do not yet have replies). When the receiver processes the request, it saves the identifier and adds the request's identifier to the reply.

When the caller processes the reply, it uses the request identifier to correlate the request with the reply. This is called a correlation identifier because of the way the caller uses the identifier to correlate each reply with the request.

A correlation ID is usually put in the header of a message. The ID is not a part of the command or data the caller is trying to communicate to the receiver.

The receiver saves the ID from the request and adds it to the reply for the caller's benefit. Since the message body is the content being transmitted between the two systems, and the ID is not a part of that, the ID is added to the header.

In the request message, the ID can be stored as a correlation ID property or simply a message ID property. When used as a correlation ID, this can cause confusion about which message is the request and which is the reply. If a request has a message ID but no correlation ID, then a reply has a correlation ID that is the same as the request's message ID.

The Correlation ID format used internally by AquaLogic Service Bus is compatible with WebSphere MQ and works with target services that are using MQ native interfaces.

The outbound transport handles asynchronous request-response messages. That is, the message flow, except for the `$outbound` transport specific data, does not distinguish between JMS request-response and HTTP request-response.

When you define a JMS request-response business or proxy service, you must first choose a design pattern. To do this, select "is response required" in the AquaLogic Service Bus Console when you design a JMS proxy or business service, then select one of the following correlation patterns on the JMS Transport Configuration page:

- JMS Correlation ID—the default pattern
- JMS Message ID

The following sections discuss these patterns. To compare the two patterns, see Comparison of Message ID and Correlation ID Patterns.

# JMS Message ID Pattern

When you create a business service using the JMS Message ID pattern, instead of defining the `Response URI`, specify the queue to be used for responses for each Managed Server in the AquaLogic Service Bus cluster. These queues must be collocated with the request queue for the service. The proxy service uses this information to set the `JMSReplyTo` property when invoking the business service so that the response is processed by the Managed Server that issued the request.

When you define a proxy service using the JMS Message ID pattern, you need not define the `ResponseURI` because the proxy service replies to the queue specified in the `JMSReplyTo` property. However, you can enter the JNDI name of the JMS connection factory for the response message.

**Note:** By default, the connection factory of the request message is used. This is useful when you use a non-XA connection factory for JMS responses, but have an XA connection factory for the request.

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.

The invoked service must copy the message ID from the request (the value of the JMS header field `JMSMessageID`) to the correlation ID of the response (setting the JMS header field `JMSCorrelationID`). In addition, the invoked service must reply to the queue specified in the `JMSReplyTo` header field.

If you choose the JMS Message ID Pattern, the response arrives at the appropriate managed node.

A JMS proxy service using this pattern can be used in a cluster without further configuration. A JMS business service is available in a cluster: however, when a Managed Server is added to the cluster, all the business services become invalid. To correct this, ensure that the number of response queues equals the number of Managed Servers that specify the JMS Message ID correlation pattern in the AquaLogic Service Bus cluster.

**Note:** The JMS Message ID correlation pattern is not supported when a proxy service invokes another proxy service.

# JMS Correlation ID Pattern

When you design a business service in Java, make sure that you set the value of JMS Correlation ID on the response to the value of JMS Correlation ID on the request before sending the JMS

response to a queue. For more information on configuring business and proxy services, see Business Services and Proxy Services in *Using the AquaLogic Service Bus Console*.

You can obtain the JMS Correlation ID when you receive a message using:

```
String getJMSCorrelationID()
```

The above method returns correlation ID values that provide specific message IDs or application specific string values.

To set the JMS Correlation ID() when you send a message:

```
void setJMSCorrelationID(String correlationID)
```

# Comparison of Message ID and Correlation ID Patterns

The JMS request-response patterns differ in the following ways:

- The method by which the response is correlated with the request
- The choice of the response queue

The differences between these two patterns are summarized in Table 2-1

**Table 2-1 Differences Between Message ID and Correlation ID Patterns**

| JMS Pattern Name | Response Queue | CorrelationID |
|---|---|---|
| Correlation ID Pattern | All responses go to the same fixed queue. | The server copies the request Correlation ID to the response Correlation ID. |
| Message ID Pattern | The responses dynamically go to the queue indicated by the `JMSReplyTo` property. | The server copies the request Message ID to the response Correlation ID. |

When the Correlation ID pattern is used, the service that is invoked responds to a fixed queue. The response always arrives on the same queue and the client has no control over the queue to which the response arrives. For example, if 10 clients send a message, they all get the response to the same queue.

Therefore, clients must filter the messages in the response queue to select the ones that pertain to them. Filtering criteria are configured in the request message Correlation ID property, and the server is configured to echo this to the response Correlation ID property.

In the case of Message ID pattern, the client's JMSReplyTo property tells the server where the response should be sent. This queue is specific to the client's server and hence responses to different clients will go to different queues. The server sets the JMS Correlation ID of the response to the JMS ID of the request.

Correlation by MessageID is commonly used by many IBM MQ applications as well as JMS applications and is the standard method to correlate request and response.

If you have multiple WebLogic client domains invoking a target WebLogic domain using JMS request-response, with the Message ID pattern, you can set up both the request and response queues as SAF queues. However, this is not possible with the Correlation ID pattern that uses a single queue for all the responses.

The Correlation ID pattern has two major advantages:

- The response queue configuration is simple and it need not change every time a new Managed Server is added to the AquaLogic cluster.
- Correlation ID can also be used in cases where a proxy service in the AquaLogic domain needs to invoke another proxy service in the same domain.

# Interoperating with JAX-RPC over JMS

Workshop for WebLogic Platform 9.2 allows you to create JAX-RPC Web Services that use JMS transport, in addition to HTTP-HTTPS. These JMS transport JAX-RPC Web Services use a JMS queue as the mechanism for retrieving and returning values associated with operations. You can use the JMS Message ID pattern to invoke a JMS transport JAX-RPC Web service.

You can also invoke a JMS Request-Response AquaLogic Service Bus proxy service from a JAX-RPC static stub, which the WebLogic Platform 9.2 clientgen Ant task generates.

This section includes the following topics:



## Invoking a JAX-RPC Web Service Using the JMS Message ID Pattern

To invoke a JMS transport JAX-RPC Web Service using the JMS Message ID pattern, complete the following steps:

1. Create a JMS Request-Response AquaLogic Service Bus business service that uses the JMS Message ID pattern to invoke the JMS transport JAX-RPC Web Service.

   This business service uses JMS transport. The JMS queue JNDI name portion of the end point URI must be the same as the queue attribute specified in the `@WLJmsTransport` annotation of the JMS transport JAX-RPC Web Service. For example:

   ```
   jms://localhost:7001/AJMSConnectionFactoryJNDIName/JmsTransportServiceRequestQueue
   ```

   The JNDI name of the JMS queue (or queues) assigned to the Destination field, in the Response JNDI Names area, must be associated with a JMS server targeted at the WebLogic Server name that is displayed in the Target field.

2. Create an AquaLogic Service Bus proxy service that contains a Routing (or Service Callout) action to the JMS Request/Response business service that you created in step 1.

   The Request Actions area of the Routing action must contain a Set Transport Headers for the Outbound Request action. When you configure the Transport Headers action, you must add two JMS headers for the Outbound Request action. For detailed instructions about how to configure a Transport Headers action, see "Transport Headers" in Proxy Services Actions in *Using the AquaLogic Service Bus Console*.

   In brief:

   a. Configure a Transport Headers Action by selecting Other in the Add Header field and entering a URI in the field provided.

   b. Select Set Header to <Expression> and create the expression by entering a concatenation of the values specified for the `contextPath` and `serviceUri` attributes (in the `@WLJmsTransport` annotation of the JMS transport JAX-RPC Web Service), preceded by a forward-slash. For example, you have the following `@WLJmsTransport` annotation:

      ```
      @WLJmsTransport(
      contextPath="transports",
      serviceUri="JmsTransportService",
      portName="JmsTransportPort",
      queue="JmsTransportServiceRequestQueue"
      )
      ```

      You would enter the following expression in the XQuery Text input area when you configure the Transport Headers:

      ```
      /transports/JmsTransportService
      ```

c. To specify the second JMS Header, select Other in the Add Header field again, and enter `_wls_mimehdrContent_Type` in the associated field.

d. Select Set Header to <Expression> and enter `text/xml; charset=UTF-8` in the XQuery Text input area.

## Invoking a JMS Request-Response Proxy Service from a JAX-RPC Client

For a scenario in which a JAX-RPC WebLogic Server client invokes a proxy service, you must set the _wls_mimehdrContent_Type JMS header for the proxy service's inbound response.

You must specify the header when you issue the response to the incoming JMS Message ID Pattern request.

For example, for the scenario in which you have a JAX-RPC client calling an AquaLogic Service Bus proxy service, which subsequently calls a WebLogic Server Web service, the route node configuration is as follows:

For the Request Pipeline:

1. Set the transport header for Web service context 'URI' (for example: `interop/AllocJmsDocLit`).
2. Set the transport header for _wls_mimehdrContent_Type with text/xml; charset=UTF-8.
3. Select Outbound request from the Set Transport headers menu items.
4. Enable Pass all Headers through Pipeline.

For the Response Pipeline:

1. Add an empty transport header and select Inbound response from the Set Transport headers menu.
2. Enable Pass all Headers through Pipeline.

# JMS Message ID Pattern Examples

The following examples describe the different methods by which the JMS Message ID pattern can be used.

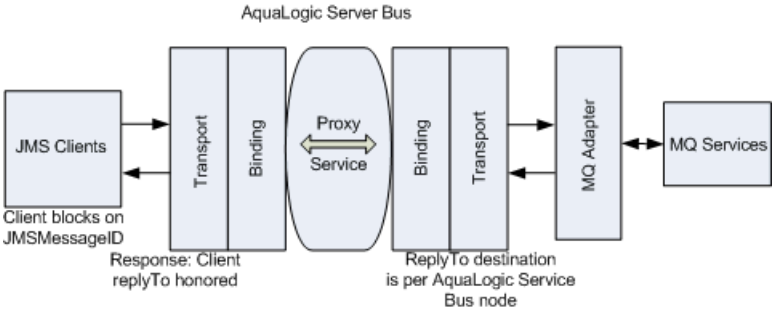
- "Example 1: An MQ Service Uses a JMS Message ID as a Correlator of the Request-Response Message" on page 2-9

## Example 1: An MQ Service Uses a JMS Message ID as a Correlator of the Request-Response Message

In Figure 2-1, the server that hosts the MQ service in the request-response communication echoes the request message ID to the response correlation ID, and sends the response to the `replyTo` queue. The response travels back and is correlated using the JMS MessageID. The AquaLogic Service Bus `replyTo` destination is set, one per AquaLogic Service Bus node in a cluster, when the business service is configured. A JMS or MQ native client can also invoke a JMS request-reply proxy service using the JMS Message ID pattern. The client needs to set the `replyTo` property to the queue where it expects the response.

The key to supporting this use case is that JMS Message ID is the expected correlator of the request-response message. You also need to create as many MQ series outbound response queues as there are cluster servers.

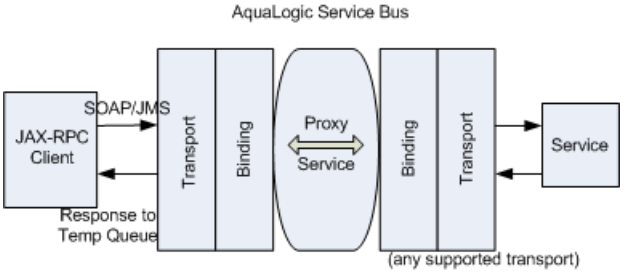
**Figure 2-1 MQ Service Uses a JMS Message ID as a Correlator of the Request/Response Message**




## Example 2: A JAX-RPC Client with AquaLogic Service Bus Proxy Service

Figure 2-2 represents a JAX-RPC client sending a message to an AquaLogic Service Bus proxy service-that is the JAX-RPC inbound case. The JAX-RPC stack employs a temporary queue to receive the response. The AquaLogic Service Bus JMS transport honors this temporary queue during run time.

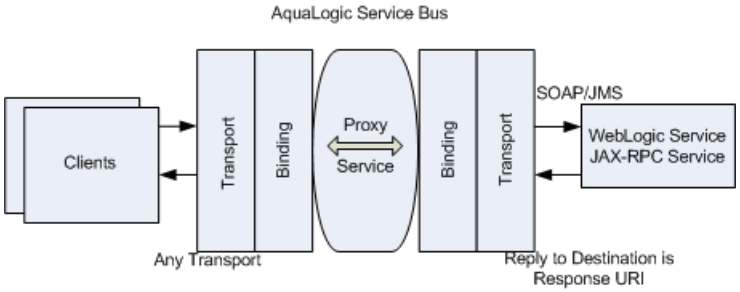
**Figure 2-2 JAX-RPC Client with AquaLogic Service Bus Proxy Service**




## Example 3: AquaLogic Service Bus as a Client of a WebLogic Server JAX-RPC Request/Response Service

Figure 2-3 represents the JAX—RPC outbound case or the interoperability of a WebLogic Server JAX—RPC request/response service with an AquaLogic Service Bus proxy service.

**Figure 2-3 AquaLogic Service Bus as a Client of a WebLogic Server JAX-RPC Request/Response Service**




**Note:** When a proxy service in one WebLogic Server domain needs to send a message to a proxy service in a second domain, the message must first be routed to a pass-through business service in domain 1. JMS Store and Forward between domain 1 and domain 2 forwards the inbound request message to the proxy service in domain 2. When you use JMS request/response, you can choose to forward the inbound response message from domain 2 to domain 1 using JMS Store and Forward as well. In the latter case, exported inbound request and imported inbound response queues must be configured in domain 2 for the proxy service in domain 2. Pay close attention to the JMS Store and Forward configuration.