Oracle® Service Bus

Interoperability Solutions for JMS

10*g* Release 3 (10.3)

October 2008

ORACLE®

Oracle Service Bus Interoperability Solutions for JMS, 10*g* Release 3 (10.3)

Contents

3. Using the JMS Transport

Oracle Service Bus Interoperability with JMS

The following sections describe features and concepts related to interoperability between Oracle Service Bus and WebLogic JMS, and between Oracle Service Bus and WebSphere MQ:

- 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

- Interoperability with WebSphere MQ

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.

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

- WebLogic Server 9.x and 10.0 JMS

- IBM WebSphere MQ/JMS 5.3

- TIBCO Enterprise Message Service 4.2

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

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

For information about WebLogic Server 10.0 JMS, see:

- Managing Your Applications in *Programming WebLogic JMS*

- Configure JMS Servers in the *WebLogic Server Administration Console Online Help*

Note: Oracle Service Bus 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 Oracle Service Bus 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.

Oracle 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. Oracle Service Bus supports SOAP messages with JMS request-response. Oracle 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 Oracle WebLogic Server 9.x and 10.0

- Interoperating with Oracle WebLogic Workshop 8.1

- Interoperating with Oracle WebLogic Server 8.1

Interoperating with Oracle WebLogic Server 9.x and 10.0

When you use JMS binding to configure a service in WebLogic Server 9.x and 10.0, use the following SOAP-JMS URL format with WebLogic Server:

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

When you configure the service in Oracle 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 Oracle WebLogic Server services from Oracle 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 "Inbound and Outbound Variables" in Message Context of *Oracle Service Bus User Guide*.

When a WebLogic Server 9.x or 10.0 Web Services client invokes an Oracle 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 more information, see "Adding Transport Header Actions" in Proxy Services: Actions in *Using the Oracle Service Bus Console*.

Oracle 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.

Configuring the Response Queues for Cross-domain JMS Calls

When you configure the response queue for cross-domain JMS calls, make sure that there is a separate response queue corresponding to each requesting managed server. For example:

If there are two Oracle Service Bus clustered domains (domain A and domain B) talking to a WLS domain. Let's say the WLS domain has 2 managed servers. If domain A has 3 managed servers and domain B has 4 managed servers, you need to configure 7 distinct queues to serve as response queues $(3 + 4 = 7)$ on the WLS domain for sending responses back to domain A and domain B. These queues could be distributed queues (with members on both the managed servers of the WLS domain).

Note: When the JMS requests come from multiple proxy services hosted by different remote domains, you must configure the back-end domain hosting the JMS business service with the separate sets of response queues corresponding to each requesting domain.

Interoperating with Oracle WebLogic Workshop 8.1

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

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

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

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

When you invoke Oracle WebLogic Workshop services from Oracle 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. For more information, see "Adding Transport Header Actions" in Proxy Services: Actions in *Using the Oracle Service Bus Console.*

When a WebLogic Workshop client (for example, a control) invokes an Oracle 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.

Oracle 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 of *Oracle Service Bus User Guide*.

Interoperating with Oracle 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 Oracle Service Bus JMS URL format shown in the preceding sections. When invoking Oracle WebLogic Server services from Oracle 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 Oracle 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 more information, see "Adding Transport Header Actions" in Proxy Services: Actions in *Using the Oracle Service Bus Console.*

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

Naming Guidelines for Domains, WebLogic, and JMS Servers

If you are working with more than one domain, make sure that 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, Oracle Service Bus 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, Oracle 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 Oracle Service Bus in the *Oracle 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.oracle.com/test/">
   ...
   <types>
      <xsd:schema targetNamespace="http://www.oracle.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 Oracle Service Bus 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.oracle.com/test/

  If the prefix test is not already known to Oracle Service Bus, you must declare it.

2. Configure a reply action with failure.

For information about configuring Reply Actions in the Oracle Service Bus Console, see Proxy Services Actions in *Using the Oracle 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 *WebLogic Web Services: Reference*.

Interoperability with WebSphere MQ

Oracle Service Bus connects to WebSphere MQ via the WebSphere MQ JMS interface. That is, Oracle Service Bus is a WebSphere MQ JMS client.

WebSphere MQ can interface with Oracle Service Bus in two ways:

- Oracle Service Bus acts as the front-end of WebSphere MQ to accept service requests from other applications and converts them to WebSphere MQ requests.

- WebSphere MQ sends messages to other applications through Oracle Service Bus.

For more information, see Using the WebSphere JMS MQ Interface in *MQ Transport User Guide*.

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 section describes the Message ID and Correlation ID patterns supported in Oracle Service Bus for JMS request-response and describe how Oracle Service Bus uses these patterns to interoperate with Java API for Remote Procedure Call (JAX-RPC) web services. Examples are also provided.

The following topics are included:

- Overview of JMS Request-Response and Design Patterns

- JMS Message ID Pattern

- JMS Correlation ID Pattern

- Comparison of Message ID and Correlation ID Patterns

- Interoperating with JAX-RPC over JMS

- JMS Message ID Pattern Examples

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 Oracle Service Bus, each reply message should contain a unique identifier called the correlation ID, which 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 that 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 Oracle 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 the **Is Response Required** check box in Oracle 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

Note: JMS request/response messaging does not have reliable responses because the mapping of the correlation ID to the proxy service is stored in memory. If there is a failure/restart in between sending the request and receiving the response, the response will be discarded. One possibility for some situations is to not use JMS request/response, but to use two JMS 1-way proxies. Use a 1-way JMS proxy for delivering the message and a second 1-way JMS proxy in the reverse direction for delivering the response.

For information about creating JMS proxy and business services, see Proxy Services: Creating and Managing and Business Services: Creating and Managing in *Using the Oracle Service Bus Console*.

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 Oracle 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 Oracle 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 information about creating JMS proxy and business services, see proxy Services: Creating and Managing and Business Services: Creating and Managing in *Using the Oracle 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 cluster.

- Correlation ID can also be used in cases where a proxy service in the domain needs to invoke another proxy service in the same domain.

Interoperating with JAX-RPC over JMS

Workshop for WebLogic 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 Oracle Service Bus proxy service from a JAX-RPC static stub, which the WebLogic clientgen Ant task generates.

This section includes the following topics:

- Invoking a JAX-RPC Web Service Using the JMS Message ID Pattern
- Invoking a JMS Request-Response Proxy Service from a JAX-RPC Client

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 Oracle Service Bus business service that uses the JMS Message ID pattern to invoke the JMS transport JAX-RPC web service. For more information, see "JMS Transport Configuration page" in Business Services: Creating and Managing in *Using the Oracle Service Bus Console*.

   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/JmsTransportServiceR
   equestQueue
   ```

   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 Oracle 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 Oracle 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 Oracle 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.

Note: For detailed instructions about how to configure a Transport Headers action, see "Transport Headers" in Proxy Services Actions in *Using the Oracle Service Bus Console*.

JMS Message ID Pattern Examples

The following examples describe the different methods by which the JMS Message ID pattern can be used.
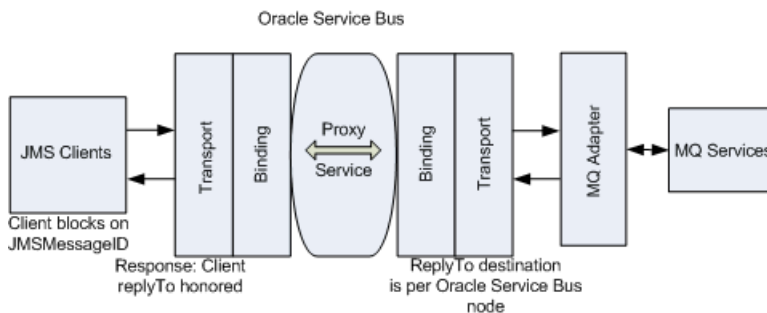
Example 1: An MQ Service Uses a JMS Message ID as a Correlator of the Request-Response Message

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 Oracle Service Bus `replyTo` destination is set, one per Oracle 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.
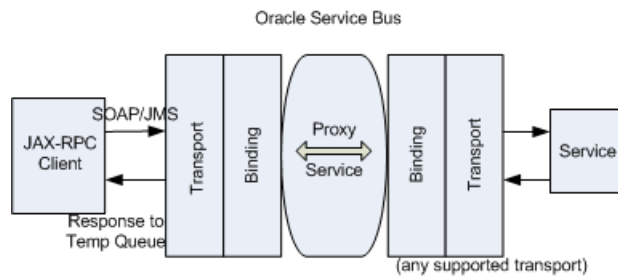
Figure 2-1 MQ Service Uses a JMS Message ID as a Correlator of the Request/Response Message

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

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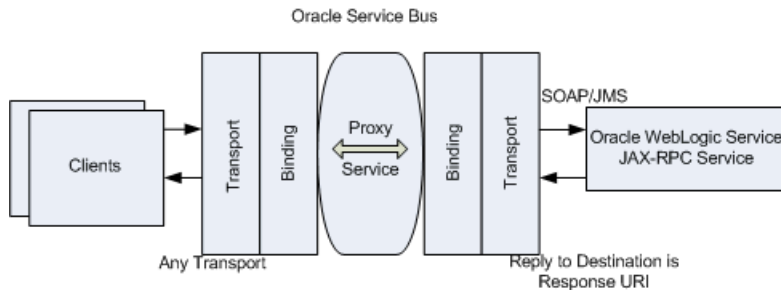
Figure 2-2 JAX-RPC Client with Oracle Service Bus Proxy Service

Example 3: Oracle 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 Oracle Service Bus proxy service.

Figure 2-3 Oracle 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.

Using the JMS Transport

You can select JMS as the transport protocol for proxy and business services of any service type (except Transport Typed Service type for business service). The proxy services and business services can be configured to use the JMS transport as described in:

- Configuring Proxy Services using JMS Transport Protocol

- Configuring Business Services using JMS Transport Protocol

For more information, see Business Services: Creating and Managing and Proxy Services: Creating and Managing in *Using the Oracle Service Bus Console.* For information about error handling for business services, see Error Handling.

When you configure a proxy service, you can use a Transport Header action to set the header values in messages. For more information, see Transport Headers.

For information about interoperability of Oracle Service Bus with WebSphere MQ, see Using the WebSphere JMS MQ Interface in *Native MQ Transport User Guide*.

In Oracle Service Bus, you can use the default dispatch policy when you configure a service or, optionally, configure custom work managers in WLS. For more information, see "Creating a Work Manager" in Using the WS Transport in *WS Transport User Guide*.

Configuring Proxy Services using JMS Transport Protocol

You can select the JMS transport protocol when you configure any type of proxy service and the endpoint URI is of the form:

`jms://<host:port[,host:port]*/factoryJndiName/destJndiName>`

where

- `host`: is the name of the system that hosts the service.

  In a cluster: The host names in the JMS URI must exactly match the host names of the cluster servers as they are configured in WebLogic Server.

- `port`: is the port number at which the connection is made.

- `[,host:port]*:` indicates that you can configure multiple hosts with corresponding ports.

- `factoryJndiName:` The name of the JNDI Connection Factory. For more information on how to define a connection factory queue, see Configure resources for JMS system modules in *Administration Console Online Help*.

- `destJndiName`: is the name of the JNDI destination.

To target a JMS destination to multiple servers, use the following format of the URI:
`jms://host1:port,host2:port/QueueConnectionFactory/destJndiName`
where `QueueConnectionFactory` is name of the connection factory queue. For more information on how to define a connection factory queue, see Configure resources for JMS system modules in *Administration Console Online Help*.

To configure a proxy service using JMS transport protocol you must specify values for the following fields:

- Destination Type: You must specify the destination to be of one of the following:
  - Queue: This defines a point-to-point destination type. You can use this destination type for peer asynchronous communications. A message, which is delivered to a queue is distributed to only one destination.
  - Topic: This defines a publish or a subscribe destination type. You can use this destination type for peer asynchronous communications. A message, which is delivered to a topic is distributed to all the subscribers of the topic.

- Is Response Required: If you expect a response to the outbound message, you must specify the following parameters for the response message:

  – Response Correlation Pattern: This configures the design pattern for JMS response message to be one of the following:

    • JMS Correlation ID: You must select `JMS Correlation ID` design pattern for your message if you want to correlate by JMS Correlation ID and send the response to the URI configured in the `Response URI` field.

    Note: The `Response URI` field is active only when you select `JMS Correlation ID` design pattern for your response message.

    • JMS Message ID: You can JMS Message ID design pattern for your message if you want to correlate by JMS Message ID and send the response to the `JMSReplyTo Destination` by configuring the `Response Connection Factory` field.

    Note: The `Response Connection Factory` field is active only when you select `JMS Message ID` design pattern for your response message.

  – Response Message Type: You can set type of the response message to `Bytes` or `Text`.

  – Response Encoding: You can set the encoding for the response message. The default encoding is `UTF-8`.

  – Client Response Timeout: This parameter specifies the response timeout interval, in seconds.

- Request Encoding: You can set the encoding for the request message. The default encoding is `UTF-8`.

Dispatch Policy: This specifies the dispatch policy for the endpoint. You must configure Work Managers in the WebLogic Server Administration Console in order to have other dispatch policies in addition to the default dispatch policy. For information about work managers, see Using Work Managers to Optimize Scheduled Work and Create Work Manager in WebLogic Server *Administration Console Online Help*.

- Advanced Settings: Click the ⊗ icon to expand the Advanced Settings section. Configuring parameters in this section is optional.

  You can set the following parameters in the section:

  – Use SSL: This specifies whether the connections can be made over SSL or not.

  – Message Selector: You can use this field to specify the criteria for selecting messages.

  – Durable Subscription: You can check if the subscription is durable only if the destination type is Topic.

  – Retry Count: In this field you can configure the maximum number of retries for the connection.

  – Retry Interval: In this field you can configure the time interval in milliseconds between consecutive retries.

  – Error Destination: In this field you can configure the name of the target destination for the messages, which have reached the maximum number of retry count.

  – Expiration Policy: In this field you can specify the message expiration policy to be used when you encounter an expired message at a WebLogic Server or a JMS destination.

  – Is XA Required: This represents if the connection factory is XA or not. When enabled, and if the connection factory is available, make sure that the connection factory is of XA type.

  – Synchronous Transactional: Specifies that the response pipeline inherits the transaction from the request pipeline. This means that the transaction will not commit until both pipelines have completed. This field is enabled only when the Is Response Required field is checked.

  – JMS Service Account: In this field you can specify the name of the service account resource to be used to make the connection over the secured socket layer. The same service account is used for both JNDI and JMS purposes.

For more information about configuring proxy services using JMS transport, see JMS Transport Configuration Page in Proxy Services: Creating and Managing in *Using the Oracle Service Bus Console*.

Transport Headers

The various headers related to the JMS transport are listed in Table 3-1. All the headers except the Unit of Order header are common to both outbound requests and inbound response.

Table 3-1 JMS Transport Headers

Header	Description
JMSMessageID	The JMSMessageID header field contains a value that uniquely identifies the message sent by a provider.
JMSDeliveryMode	The JMSDeliveryMode header field contains the delivery mode specified when the message was sent.
JMSExpiration	This header contains the expiration time of the message that is calculated as the sum of the time-to-live value specified on the send method and the current GMT value.
JMSPriority	The JMSPriority header field contains the priority of the message. When a message is sent, this field is ignored. After the send operation is complete, the field holds the value that is specified by the method sending the message.
JMSType	The JMSType header field contains the message type identifier that is specified by a client when a message is sent.
JMSCorrelation ID	This is used to link one message with another. For example to link a request message with a response message.
JMSXAppID	This is one of the JMS defined properties that specifies the identity of the application that sends the message. This is set by the JMS provider
JMSXGroupID	This one of the properties defined by JMS that specifies the group id of the message group to which the message belongs. This is set by the client
JMSXGroupSeq	This one of the properties defined by JMS that specifies the sequence of the message inside the message group.
JMS_IBM_Format	This contains the nature of the application data. For more information, see JMS_IBM_FORMAT.
JMS_IBM_Report_Except ion	Request exception reports. This also specifies how much of the application data from the message must be retained in the report message. For more information, see JMS_IBM_Report_Exception.

Table 3-1 JMS Transport Headers

Header	Description
JMS_IBM_Report_Expiration	Request expiration reports. This also specifies how much of the application data from the message must be retained in the report message. For more information, see JMS_IBM_Report_Expiration.
JMS_IBM_Report_COA	Request a confirm on arrival reports. This also specifies how much of the application data from the message must be retained in the report message. For more information, see JMS_IBM_Report_COA.
JMS_IBM_Report_COD	Request a confirm on delivery reports. This also specifies how much of the application data from the message must be retained in the report message. For more information, see JMS_IBM_Report_COD.
JMS_IBM_Report_PAN	Request a positive action notification reports. For more information, see JMS_IBM_Report_PAN.
JMS_IBM_Report_NAN	Request a positive action notification reports. For more information, see JMS_IBM_Report_NAN
JMS_IBM_Report_Pass_Msg_ID	Request that the message identifier of any report or reply message is the same as that of the original message. For more Information, see JMS_IBM_Report_Pass_Msg_ID.
JMS_IBM_Report_Pass_Correl_ID	Request that the correlation identifier of any report or reply message is the same as that of the original message. For more information, see JMS_IBM_Report_Pass_Correl_ID.
JMS_IBM_Report_Discard_Msg	Request that the message is discarded if it cannot be delivered to its intended destination. For more information, see JMS_IBM_Report_Discard_Msg.
JMS_IBM_MsgType	The type of a message. For more information, see JMS_IBM_MsgType.
JMS_IBM_Feedback	This indicates the nature of a report message. For more information, see JMS_IBM_Feedback.
JMS_IBM_Last_Msg_In_Group	Indicates whether the message is the last message in a message group. For more information, see JMS_IBM_Last_Msg_In_Group.
JMS_BEA_UnitOfOrder	**Note:** This header is valid only for the Inbound Response.

Configuring Transport Headers

You can configure the transport headers for both inbound and outbound requests in the Message Flow. For information about the transport headers related to the JMS transport, see Transport Headers.

In the pipeline, use a Transport Header action to set the header values in messages. For information about adding transport header actions, see Adding Transport Header Actions in *Using the Oracle Service Bus Console*.

Note: For information about the limitations for JMS transport headers, see Understanding How the Run Time Uses the Transport Settings in the Test Console in *Using the Oracle Service Bus Console* and "Limitations to Transport Header Values You Specify in Transport Header Actions" in Modelling Message Flows in *Oracle Service Bus User Guide*.

Configuring Business Services using JMS Transport Protocol

You can select the JMS transport protocol when you configure any type of business service and the endpoint URI is of the form:

`jms://<host:port[,host:port]*/factoryJndiName/destJndiName >`

where

- `host`: is the name of the system that hosts the service.

  In a cluster: The host names in the JMS URI must exactly match the host names of the cluster servers as they are configured in WebLogic Server.

- `port`: is the port number at which the connection is made.

- `[,host:port]*:` indicates that you can configure multiple hosts with corresponding ports.

- `factoryJndiName:` The name of the JNDI Connection Factory. For more information on how to define a connection factory queue, see Configure resources for JMS system modules in *Administration Console Online Help*.

- `destJndiName`: is the name of the JNDI destination.

To target a target a JMS destination to multiple servers, use the following format of the URI:

`jms://host1:port,host2:port/QueueConnectionFactory/destJndiName`

where *QueueConnectionFactory* is name of the connection factory queue. For more information on how to define a connection factory queue, see Configure resources for JMS system modules in *Administration Console Online Help*.

When you register a JMS business service, you must manually edit the URI from the WSDL file when adding it to the service definition. The URI format is as follows:

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

To configure a business service using the JMS transport protocol you must specify values for the following fields:

- Destination Type: You can specify the destination to be of one of the following:
  - Queue: This defines a point-to-point destination type. You can use this destination type for peer asynchronous communications. A message, which is delivered to a queue is sent to only one destination.
  - Topic: This defines a publish or a subscribe destination type. You can use this destination type for peer asynchronous communications. A message, which is delivered to a topic is distributed to all the subscribers of the topic.

- Is Response Required: You can specify if you expect a response to the outbound message. If you expect a response you must specify the following parameters for the response message:
  - Response Correlation Pattern: You can configure the design pattern for JMS response message to be one of the following:
    - JMS Correlation ID: You must select a JMS Correlation ID design pattern for your message if you want to correlate by JMS Correlation ID and send the response to the URI configured in the Response URI field.

      Note: The Response URI field is active only when you select JMS Correlation ID design pattern for your response message.

    - JMS Message ID: You must select JMS Message ID design pattern for your message if you want to correlate by JMS Message ID and send the response to the JMSReplyTo Destination by configuring the Response Connection Factory field.

      Note: The Response Connection Factory field is active only when you select JMS Message ID design pattern for your response message.

- – Response JNDI Names: You can specify the JNDI names of the response destinations. You need to configure one for each server in the domain.

- – Response Message Type: You can set type of the response message to `Bytes` or `Text`.

- – Response Encoding: You can set the encoding for the response message. The default encoding is `UTF-8`.

- – Response Timeout: The amount of time to wait for a response, the response timeout interval, in seconds.

- Request Encoding: You can set the encoding for the request message. The default encoding is `UTF-8`.

- Dispatch Policy: You can specify the dispatch policy for the endpoint. You must configure Work Managers in the WebLogic Server Administration Console in order to have other dispatch policies in addition to the `default` dispatch policy. For more information on how to configure a Work Manager, see Create a Global Work Manager in WebLogic Server Administration Console.

- Advanced Settings: To configure the advance settings click on ⊗ icon on the right hand side to expand the advanced settings section. Configuring parameters in this section is optional.

  You can set the following parameters in the section:

  - – Use SSL: This specifies if the connections can be made over SSL or not.

  - – Expiration: This specifies the time interval in milliseconds after which the message will expire. Default value is `0`, which means that the message never expires.

  - – Enable Message Persistence: This represents the mode of delivery of the JMS message. When this option is enabled, the delivery mode is persistent else non-persistent.

  - – Unit Of Order: This is a value added feature of WebLogic, which enables a message producer to group messages into a single unit with respect to the order of processing. This single unit is called the `Unit of Order`. All the messages in a unit must be processed sequentially in the same order they were created.

    Note: This is supported by WebLogic Server 9.0.

  - – Pass Caller's Subject: This specifies that Oracle Service Bus is to pass the authenticated subject when sending a message. When you enable this field and the business service targets JMS resources in a different domain, enable global trust on both domains. See Configuring Security for a WebLogic Domain in *Securing WebLogic Server*.

–

  – JMS Service Account: You can select the service account resource to be used for the JMS server connection. The same service account is used for both JNDI and JMS purposes.

Note: Configuring a JMS request-response application with response correlation by JMS ID for a business service you must:

  - Creating UDQs and local queues targeted to one chosen JMS server.

  - Disable the default targeting for the UDQs that deploys the UDQ on the cluster and creates the member queues on all JMS.

Error Handling

You can configure JMS-transport business services to handle application and communications errors as follows:

- Application errors

  You can specify whether or not to retry business service endpoint URIs when application errors occur. For more information, see `Retry Application Errors` option in Creating and Configuring Business Services - Transport Configuration page in *Using the Oracle Service Bus Console*.

  An application error occurs when for a JMS business service configured with request/response, the `System.getProperty("com.bea.wli.sb.transports.jms.ErrorPropertyName", "SERVER_ERROR")` property is `true` in the response message. In this scenario, the JMS transport provider calls the error method with the `TRANSPORT_ERROR_APPLICATION` error code.

- Communication errors

  You can configure business service URIs to be taken offline when such communication errors occur. When you configure the operational settings for the business service, you can enable the business service endpoint URIs to be taken offline after the specified retry interval. For more information, see "Configuring Operational Settings for Business Services" and "Viewing Business Services Endpoint URIs Metrics" in Monitoring in *Using the Oracle Service Bus Console*.

Connection errors occur when any JMS exceptions or naming exceptions occur during any of the following activities:

– Looking up a JMS connection factory.

– Creating a JMS connection from a JMS connection factory.

– Creating a JMS session using a connection object.

– Looking up of a JMS destination.

– Creating a sender from the session object.

– Sending a JMS message using the sender and destination object.