BEA AquaLogic Service Bus™

Interoperability Solutions for Web Services Remote Portlets (WSRP)

Version: 2.6
Document Revised: January 2007

Contents

Introduction

Configuring AquaLogic Service Bus for WSRP

WSRP Interoperability Example

CHAPTER 1

Introduction

Web Services for Remote Portlets (WSRP) is a mechanism used to generate markup fragments on a remote system for display in a local portal application. This mechanism is gaining popularity in recent years. This chapter describes how AquaLogic Service Bus can be used to provide Service Level Agreement (SLA) monitoring in applications that use WSRP.

This section discusses the following topics:

WSRP Producers and Consumers

WSRP involves two integral components:

- The remote application, called a *WSRP producer* (referred to as a *producer* in this section) implements standards-based Web Services using the SOAP specification over HTTP. Producers are created using WebLogic Portal or third-party implementations of WSRP.
- A *WSRP consumer* (referred to as a *consumer* in this section) is a Portal application. Typically, the consumer application references the producer's WSDL when the portal is designed, and the consumer directly accesses the producer.

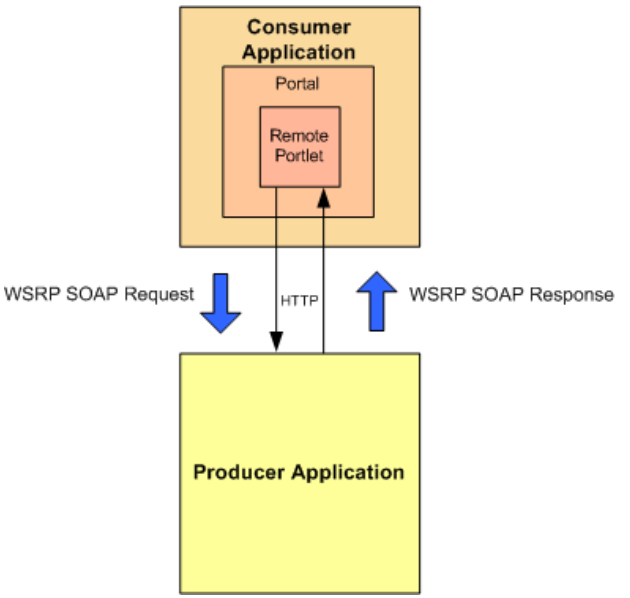
WSRP Architecture

This section describes basic WSRP architecture and shows how this architecture can be enhanced by adding AquaLogic Service Bus.

Basic Architecture

Figure 1-1 shows the basic WSRP SOAP request and response flow between a producer application and a consumer application.

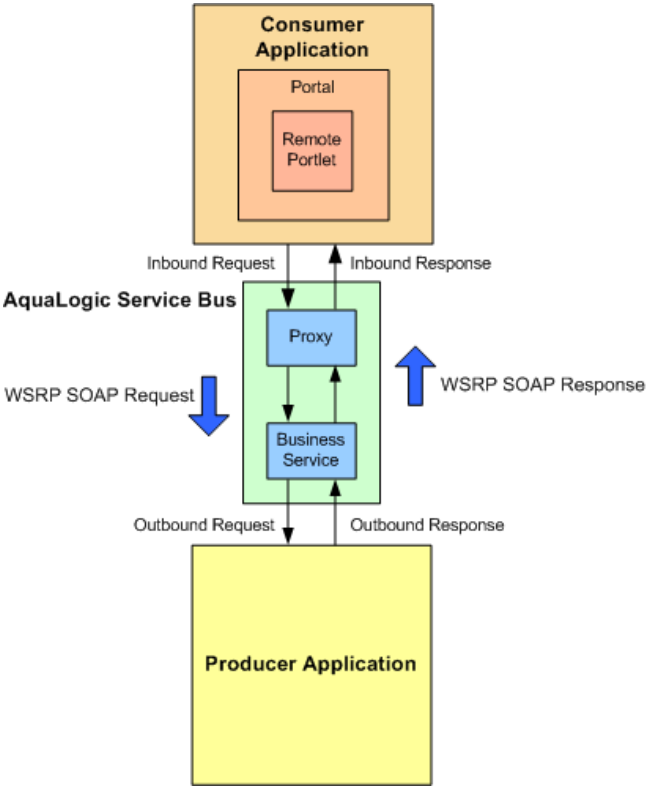
Figure 1-1 Basic Request/Response Flow Between Producer and Consumer Applications



Enhanced Architecture with AquaLogic Service Bus

Because a WSRP producer implements SOAP Web Services, an enterprise service bus (such as AquaLogic Service Bus) can be used as an intermediary between the producer and consumer to provide Service Level Agreement (SLA) monitoring, as shown in Figure 1-2.

Figure 1-2 Enhanced WSRP Request / Response Flow Via AquaLogic Service Bus



In this architecture, the WSRP SOAP request/response flow occurs in the following sequence:

1. **Inbound Request:** The consumer calls the proxy service in the AquaLogic Service Bus.
2. **Outbound Request:** The proxy service routes the request (a message containing the SOAP body and transport headers) to the producer.
3. **Outbound Response:** The producer returns a reply to AquaLogic Service Bus.
4. **Inbound Response:** The proxy service returns the reply (a message containing the SOAP body and transport headers) to the consumer.

The remainder of this section provides instructions for configuring the AquaLogic Service Bus to proxy service requests for WSRP services. It describes services that a producer provides, along with other attributes of WSRP that must be used to properly configure AquaLogic Service Bus.

This section also discusses different possible strategies that can be used to monitor producers with increasing degrees of detail. Finally, it discusses load balancing and failover with WSRP.

WSRP Design Concepts

This section describes the following WSRP design concepts:

- WSRP WSDLs
- WSRP Messages

WSRP WSDLs

Table 1-1 describes various kinds of services offered by producers.

Table 1-1 Producer Services

Service	Description
Service Description	*Required service.* Describes the producer and the portlets that the producer makes available to consumers.
Markup	*Required service.* Manages user interaction with a remote portlet and returns the HTML markup used to render the portlet.
Registration	*Optional service.* Required for complex producers. Allows consumers to register themselves with the producer.
Management	*Optional service.* Provided by complex producers for managing portlet customization and portlet preferences.
Markup Extension	Provided by BEA Portal producers and replaces the Markup service. Markup Extension allows more efficient message handling by using multipart MIME messages for transmitting HTML markup content.

Each producer implements a minimum of two services (Service Description and Markup). A *simple producer* offers just these two services. A *complex producer*, however, provides two additional services (Registration and Management). WebLogic Portal producers also implement an extension service (Markup Extension) that replaces the standard Markup service.

These services are described using a standard WSDL format. The producer supplies a single URL for retrieving its WSDL, which describes all the services that are available from that producer.

The end points for each service indicate whether the consumer should use transport-level security (HTTP(s)) or not to communicate with the producer.

WSRP Messages

WSRP uses SOAP over HTTP for all messages sent between producers and consumers. In addition to using standard message formats in the SOAP Body, WSRP requires that consumers must set at least a `SOAPAction` header, cookie headers, and the usual HTTP headers (such as `Content-Type`). Producers will return a session cookie, plus any application-specific cookies, in the HTTP transport header of the response message. The consumer must return the session cookie in subsequent request messages.

CHAPTER 2

Configuring AquaLogic Service Bus for WSRP

The AquaLogic Service Bus Console, which is described in the *Using the AquaLogic Service Bus Console*, is used to configure AquaLogic Service Bus. For more information about creating WSRP-enabled portals using WebLogic Portal, see *Federated Portals Guide*.

Configuring AquaLogic Service Bus for WSRP involves the following tasks:

- Implementing a service that consumers can invoke to obtain an appropriate WSDL for a particular producer.
- Implementing the details of conveying a consumer's request to the producer and returning the response to the consumer.

This chapter describes the following tasks:

Getting the Producer WSDL

As a common practice, consumers contact a producer directly to obtain its WSDL. However, if AquaLogic Service Bus is used as a proxy service, then all access to the producer occurs via AquaLogic Service Bus. Therefore, a proxy service must be implemented for consumers that calls the producer's real URL to obtain its WSDL, and then transform the results as follows:

- Rewrite the endpoint address for the producer to refer to the Service Bus IP address and port
- Change the endpoint URI to refer to the AquaLogic Service Bus proxy service that reflects the required monitoring granularity (as described in "Monitoring WSRP Applications" on page 2-3)
- Change the endpoint protocol and port if transport security is used between the consumer and the AquaLogic Service Bus proxy service

The developer who creates a producer can specify whether the producer requires SSL or not (`"secure=true"`). In addition, the AquaLogic Service Bus administrator can change the security requirement to the consumer via AquaLogic Service Bus configuration. For example, if a producer does not require SSL, the AquaLogic Service Bus administrator can require consumers to use SSL by:

- Changing the WSDL to specify HTTP(s)
- Configuring the proxy services for WSRP to use HTTP(s)

When configured in this way, AquaLogic Service Bus automatically bridges the secure messages from the consumer to the non-secure messages used by the producer.

Routing Messages Between the Consumer and the Producer

After retrieving a copy of the WSDL, the consumer uses the WSDL definitions to formulate service requests and sends them to the producer via AquaLogic Service Bus. The WSRP request/response process involves the following steps:

1. The consumer sends a message to the AquaLogic Service Bus proxy service corresponding to the producer service.
2. The proxy service executes a simple message flow that routes the message (unchanged) to the actual producer service.
3. The producer formulates a response that it then sends to the consumer via AquaLogic Service Bus.
4. The consumer receives the response (unchanged) from the producer.

WSRP Web services expose portlets and those can rely on HTTP cookies and sessions. Therefore, WLSB must be configured to propagate HTTP transport headers (such as

SOAPAction and cookies). However, by default, AquaLogic Service Bus does not pass transport headers from the proxy service to the business service, because the proxy service may or may not use the same transport as the business service. Therefore, the message flow must be configured to copy the request headers from the inbound request to the outbound request. Similarly, the response headers from the business service must be copied back to the proxy service's response to the consumer.

Although it is possible to copy *all* transport headers between the proxy service and the business service, it is necessary to be more selective to avoid errors. The Set-Cookie and Cookie headers must be copied. Because AquaLogic Service Bus is the entity that assembles the final message to send, it must own some headers (such as Content-Length). For example, if the message flow were to copy the Content-Length header from the proxy service to the business service, it might result in an error because the length of the message could change during processing. Therefore ALSB must own this header.

Monitoring WSRP Applications

Monitoring tracks the usage of a producer's individual services and operations. The message flow for WSRP services introduces very little overhead, and the mapping between proxy services and producers, and between business services and producers, is simple to configure. Therefore, to satisfy SLA requirements, it is sufficient to monitor only the proxy services.

Monitoring for Proxy Services

To configure monitoring for WSRP proxy services, create a proxy service for each of the services implemented by the producer.

- Simple producers require only two proxies—one for the Markup service and one for the Description service.
- Complex producers require these two proxies and two additional proxy services for Registration and Management.

These proxy services should be based on the standard WSRP WSDLs using SOAP bindings. Only a single business service for the producer should be created, and it should be configured to use "Any SOAP Service" instead of being based on a WSDL. The message flow between the proxies and the business service should not modify the SOAP body in any way. However, just as for all WSRP message flows, it must pass the request headers via HTTP from the client request to the actual producer. Similarly, the response HTTP headers returned by the producer must be copied back to the client in the message flow.

Monitoring for Business Services

Monitoring is required for a producer's business services. Separate business services must be created for each of the Web services described in the producer's WSDL, and the business services must be defined using the WSDL. There is a one-to-one mapping between the proxy services and the business services—an unconditional routing node is sufficient in the message flow.

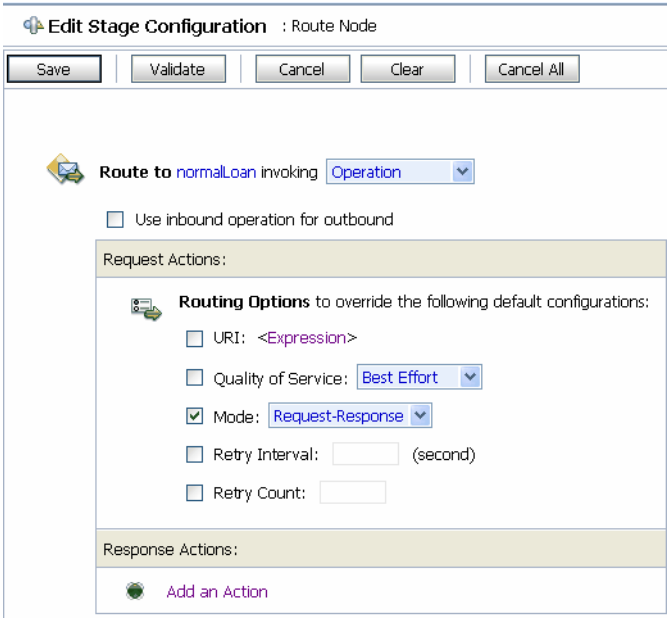
AquaLogic Service Bus requires information about which operation to use, in order that operations are counted correctly. Normally, the administrator would do this by selecting one of the operations from a drop-down menu when the business service is selected for the Route action. However, the operation specified by the client message is not the same for all messages, so a single, hard-coded value will not work here.

The administrator must ensure that the business service uses the same operation as the proxy service. While this could be achieved by specifying a Routing Table action that selects the case using the `$operation` variable, it is a very tedious approach because the WSRP standard defines 14 operations across all WSRP services, and each would require a Route action with transformations to propagate the transport headers.

An alternative when routing to the business service is, rather than selecting the operation from the drop-down menu, an administrator should use another transformation in the request actions to insert the value of `$inbound/ctx:service/ctx:operation` into `$outbound/ctx:service`.

Another alternative on the AquaLogic Service Bus console is when you are configuring the routing to a business service, select the **Use inbound operation for outbound** check box when you are editing a route node to avoid low level Xquery manipulation as in Figure 2-1.

Figure 2-1 Passing an Operation From Inbound to Outbound



With this transformation, the operation for the business service is dynamically set to the same value as was specified for the proxy service, and AquaLogic Service Bus will correctly count and monitor all operations of the service.

Load Balancing and Failover

AquaLogic Service Bus allows business services to define multiple endpoints that all provide the same Web service. When multiple endpoints are defined, AquaLogic Service Bus can automatically distribute load balance requests across endpoints, and it can automatically failover requests when an endpoint is inaccessible. However, WSRP imposes some limitations on the use of these features.

Portlets are a means of exposing a user interface to an application. Therefore, portlets typically have session data associated with them. To preserve session data, requests to the portlet must be directed to the same server (or cluster) that serviced the original request. This requirement makes load balancing via AquaLogic Service Bus inappropriate. Multiple endpoints in a business service will usually target different servers or cluster. Because there is no communication among servers that are in separate clusters, there is no way to preserve the session. Therefore, if multiple

endpoints are defined for a WSRP business service, then the load-balancing algorithm must be set to `"none"`.

Multiple endpoints can be used to provide redundancy in certain circumstances in the event that one of the endpoints is unavailable. The WSRP service is still available on a secondary endpoint. However, any session data that existed at the time the first endpoint failed will not be available on other endpoints.

This failover configuration is an option only for simple producers (see "WSRP WSDLs" on page 1-4), not for complex produces. Complex producers require that their consumers first register with the producer before sending service requests. The producer returns a registration handle that the consumer must include with each request to that producer. If a business service defines multiple endpoints, each endpoint provides and requires its own registration handle.

AquaLogic Service Bus is, however, stateless across requests—it does not maintain a mapping of the correct handle to send to a particular endpoint. In fact, it would only send the registration request to a single endpoint, so the consumer would be registered with only that one producer. If that one producer is unavailable, then AquaLogic Service Bus would route a service request to another endpoint defined for that business service, but the consumer would never have registered with that new producer, and the request would fail with an `"InvalidRegistration"` fault.

The management of registration handles requires an application outside of AquaLogic Service Bus to maintain this state data. Therefore, the registration requirement precludes defining multiple endpoints for complex producers. As simple producers do not support the registration service, a failover configuration that defines multiple endpoints in the business service is possible, although session data is lost on failover.

CHAPTER 3

WSRP Interoperability Example

The examples described in this section support AquaLogic Service Bus versions 2.0, 2.1 and 2.5 (see Example Prerequisites).

This section describes a WSRP interoperability example. It contains the following topics:

Example Prerequisites

The WSRP interoperability example assumes the following components and configuration:

- WebLogic Platform 9.2
- AquaLogic Service Bus 2.0, 2.1 and 2.5
- Sample Platform domain configured at `platform:7001`
- AquaLogic Service Bus domain configured at `alsb:7001`
- Sample Portal application consumer
- Sample producer

For an AquaLogic Service Bus configuration that supports the configuration defined in this example, see the AquaLogic Service Bus/WSRP code sample, available from the AquaLogic Service Bus code samples page on BEA dev2dev:

- For AquaLogic Service Bus 2.5 see:
  https://codesamples.projects.dev2dev.bea.com/servlets/Scarab/remcurreport/true/template/ViewIssue.vm/id/S267/nbrresults/13
- For AquaLogic Service Bus 2.0 and 2.1 see:
  https://codesamples.projects.dev2dev.bea.com/servlets/Scarab/remcurreport/true/template/ViewIssue.vm/id/S175/nbrresults/13

Example Projects and Folders

This example describes the configurations corresponding to the ALSB 2.5 dev2dev code sample.

The structure of the sample is divided into two projects—one containing common resources, and the other containing resources for the sample producer.

Table 3-1 Projects in the WSRP Interoperability Examples

Folder	Description
`wsrp`	Contains common resources that are not specific to any producer.
`operationExample`	Full example supports the most fine-grained monitoring. The folder contains producer-specified resources. See "Monitoring Example" on page 3-2.

Monitoring Example

The monitoring configuration example (in the `operationExample folder`) involves configuring AquaLogic Service Bus to monitor all services and operations of a producer.

The monitoring configuration uses both business services and proxy services that are based on the WSDLs defined by the WSRP standard. The example also defines the additional resources to describe the WSRP services and extend the message flows to support monitoring at the operation level. The rest of this section describes the tasks required to implement the monitoring configuration.

Step 1: Define WSDL Resources

Import all the WSRP WSDL definition files, along with the XML schema files on which the definitions depend. All the files are available as part of the sample code associated with this example, but the standard resource locations are listed in Table 3-2.

Table 3-2 WSDL Resource Definitions

Resource Name	Location
wsrp_v1_bindings	http://www.oasis-open.org/committees/wsrp/specifications/version1/wsrp_v1_bindings.wsdl
wsrp_v1_interfaces	http://www.oasis-open.org/committees/wsrp/specifications/version1/wsrp_v1_interfaces.wsdl
wsrp_v1_types	http://www.oasis-open.org/committees/wsrp/specifications/version1/wsrp_v1_types.xsd
wlp_wsrp_v1_bindings	$BEA_HOME/weblogic81/portal/lib/wsrp/wsrp-common.jar
wlp_wsrp_v1_types	$BEA_HOME/weblogic81/portal/lib/wsrp/wsrp-common.jar
xml	http://www.w3.org/2001/xml.xsd
wsrpWSDL	http://platform:7001/producer/producer_WSDL

Producers generated by BEA Portal extend the standard WSDLs by defining an additional port that allows messages to be sent using MIME attachments. It is necessary to define these extension resources if the producer WSDL references them. In this example, an optional task is to create a resource for the WSDL used by the producer. After creating these WSDL and XML Schema resources, edit the references in each resource to resolve the dependencies on other resources.

Step 2: Create Business Services

This monitoring example uses the WSDL bindings for each port type implemented by the producer. Because a business service can be associated with only one WSDL port or binding, a separate business service resource must be created for each. A simple producer implements only the required Markup and Service Description interfaces, while a complex producer also implements the Management and Registration interfaces. The services are created identically except for the service name and types, see Table 3-3.

Table 3-3 Business Service Configuration

Service Name	Service Type
base	WSDL port: operationExample/wsrpWSDL, port="WSRPBaseService"
desc	WSDL port: operationExample/wsrpWSDL, port="WSRPServiceDescriptionService"
mgmt	WSDL port: operationExample/wsrpWSDL, port="WSRPPortletManagementService"
reg	WSDL port: operationExample/wsrpWSDL, port="WSRPRegistrationService"
ext	WSDL port: operationExample/wsrpWSDL, port="WLP_WSRP_Ext_Service"

For each service, the required attributes are listed in Table 3-4.

Table 3-4 Service Attributes for Business Services

Name	Value	Comments
Protocol	HTTP	Or HTTP(s) if the producer was created with secure="true".
Load Balancing Algorithm	none	Must be none, or session data will be lost across requests if multiple end points are defined.
Endpoint URI	For WebLogic Platform 8.1: http://platform:7001/producer/producer/	The URL is the same for Markup, Service Description, Registration Service, and Portlet Management.

Table 3-4 Service Attributes for Business Services (Continued)

Name	Value	Comments

Endpoint URI (continued)	For WebLogic Platform 9.2 the URLs are as follows:
	• Service Description:
	`http://host*:port+/wsrpProducer/produce r/wsrp_1.0/serviceDescription`
	• Markup:
	`http://host*:port+/wsrpProducer/produce r/wsrp_1.0/markup`
	• Registration:
	`http://host*:port+/wsrpProducer/produce r/wsrp_1.0/registration`
	• Portlet Management:
	`http://host*:port+/wsrpProducer/produce r/wsrp_1.0/portletManagement`
	• Markup Extension:
	`http://host*:port+/wsrpProducer/produce r/wsrp-wlp-ext-1.0/markup`
	*`host` - Replace with the actual host name on which the application is running
	`+port` - Replace with the actual port name on which the application is running
		Multiple endpoints must be defined for WSRP producers.

Step 3: Create the Proxy Services

Proxy services in this monitoring example are configured as follows:

- The proxy services must be based on the same WSDL, just as the business services are based on a WSDL.
- One proxy service is created for each business service, but each proxy service must have a different URI.
- The configuration must specify which operation is being invoked.

To create a proxy service:

1. Create the proxy service for the base WSRP service.

   As in the earlier example, create the proxy service using the existing `operationExample/base` business service as the model. This will automatically base the proxy service on the same WSDL binding as the business service, and it will create a message flow with an unconditional route action to the business service. For the Endpoint URI, you can use any URI, such as the producer name with the port type abbreviation appended to it (for example, `/operationExampleBase`).

2. Edit the message flow to add the same transformations required to copy the request transport headers and response transport headers between the consumer and producer.
   WSRP relies on data conveyed in the transport headers to function properly. In particular, producers will return to consumers session cookies in the response headers if they expect consumers to supply session cookies in subsequent requests. Similarly, producers expect consumers to provide the requested operation in the `SOAPAction` request header.

   By default, AquaLogic Service Bus does not copy transport headers from the inbound request to the outbound request, or from the outbound response to the inbound response. The message flow must therefore propagate the required headers both in and out of the business service. Because these transformations are required for every WSRP service, it is convenient to define two common XQuery resources—one for request headers and one for response headers—that extract the correct headers.

   For request headers, use the query provided in Table 3-5.

Table 3-5 Request Header Query

Name	Value
Resource Name	`wsrp/rqstHeaders`

Xquery	`declare namespace ctx="http://www.bea.com/wli/sb/context";`
	`declare namespace tp="http://www.bea.com/wli/sb/transports";`
	`declare variable $in external;`
	`$in/ctx:transport/ctx:request/tp:headers/child::*[local-name()!="Content-Length"]`

   The rqstHeaders query extracts all transport headers (except `Content-Length`) from the `$in` variable. AquaLogic Service Bus can sometimes reformat the message body so that its length no longer exactly matches the request message. Copying the length from the

original request can result in transport errors if the body is modified (for example, reformatted).

To copy the inbound request headers to the outbound business service, add the following Replace request action to the message flow:

```
Replace ./ctx:transport/ctx:request/tp:headers in variable outbound with
xqTransform()

Replace node contents

Variable Mapping (wsrp/rqstHeaders):

in: $inbound
```

Similar to the request side, the response side defines a common XQuery resource to extract all but the `Content-Length` header from the response returned from the producer.

For response headers, use the following query provided in Table 3-6.

Table 3-6 Response Header Query

Name	Value
Resource Name	`wsrp/rspncHeaders`

Xquery	`declare namespace ctx="http://www.bea.com/wli/sb/context";`
	`declare namespace tp="http://www.bea.com/wli/sb/transports";`
	`declare variable $out external;`
	`$out/ctx:transport/ctx:response/tp:headers/child::* [local-name()!="Content-Length"]`

The following replace response action in the route node propagates the required headers:

```
Replace ./ctx:transport/ctx:response/tp:headers in variable inbound with
xqTransform()

Replace node contents

Variable Mapping (wsrp/rspncHeaders):

out: $outbound
```

3. Specify which operation to be invoked.

   Generally, in a route Action that routes to a WSDL-based service, an operation to invoke (by selecting the correct operation from the drop-down menu) is specified. However, each

WSRP port implements several operations, and so the configuration requires a routing table with a case for each operation. Each case requires the same transformations to propagate the transport headers.

Creating all of the transformations in this manner may prove to be tedious. Therefore, instead of using the drop-down menu, use another transformation to copy the operation from the proxy service to the business service. Configure this transformation by adding an Insert Action to the Request Actions of the message flow:

```
Insert $inbound/ctx:service/ctx:operation as last child of ./ctx:service
in variable outbound
```

The proxy services for the other business services can be created by repeating these steps, although a shortcut can be used to avoid recreating all of the transformations manually.

For example, to create the proxy service for the Service Description service:

1. Create a new proxy service using the existing `operationExample/base` proxy service just created as the model. Following this example, use `/operationExampleDesc` for the Endpoint URI.
2. On the Summary Page, click the edit link for General Configuration. The WSDL binding is created using the Base port, so correct that here to refer to the `WSRPServiceDescriptionService` port.
3. Edit the message flow. The route action refers to the base business service. Correct this to route to the `desc` service.

Note: Use the Transport Header action to minimize low level Xquery manipulation and simplify the configuration of a proxy service. See Transport Headers section in the AquaLogic Service Bus *Console Online Help* for details.

Step 4: Retrieve the WSDL from the Producer

Create a service that will retrieve the WSDL from the producer and transform it to hide the actual producer endpoints. In this example the proxies for each producer have a different URI. The rest of this section describes how to create the resources to retrieve the producer WSDL.

Step 4.1: Create the Business Service to Retrieve the WSDL

Create a business service to obtain the WSDL from the producer. This resource is specific to the producer, so it must be created in the operationExample project. Table 3-7 describes the properties of the business service.

Table 3-7 Business Service Configuration Properties

Name	Value	Comments
Service Name	`wsdlSvc`	Any name is allowed.
Service Type	Any XML Service	Consumers usually retrieve the WSDL from the producer using an HTTP GET request. Only XML services support GET.
Protocol	HTTP or HTTP(s)	HTTP(s)
Load Balancing Algorithm	None	None preferred
Endpoint URI	http://platform:7001/producer/producer?WSDL	Although multiple endpoints may be specified for retrieving the WSDL, doing so is of limited benefit.
HTTP Request Method	GET	

Step 4.2: Create an XQuery Expression to Construct URLs

All end point addresses in the producer's WSDL must be transformed to reflect the AquaLogic Service Bus server address and the proxy service URI values. Because each producer WSDL can have four or more ports defined, it is convenient to create an XQuery expression to simplify the construction of the endpoint locations. The XQuery expression accepts the following three string variables as input and concatenates them together to form a SOAP address element:

- **base URL** for the AquaLogic Service Bus server
- **name** to identify the producer
- **extension** used to differentiate ports for a producer

Table 3-8 provides the query definition in the wsrp project.

Table 3-8 XQuery Definition in the wsrp Project

Name	Value
Resource Name	wsrp/addr
XQuery	declare variable $baseURL external; declare variable $name external; declare variable $svc external; declare namespace soap="http://schemas.xmlsoap.org/wsdl/soap/"; <soap:address location="{concat($baseURL, $name, $svc)}"/>

Step 4.3: Create a No-Op Proxy Service

Create a service that does nothing. To create this service, define a new proxy service in the wsrp project folder with the resource name nullSvc. Accept all of the defaults for this service. Configuring this proxy service creates a message flow for the service of an echo node.

Step 4.4: Create a Common Proxy Service to Retrieve the WSDL

Create a proxy service used by consumers to get WSDLs from producers. This proxy service is appropriate for any producer configuration modeled on this sample. The example described in this section is only a suggestion—a different approach might the specific requirements of a given implementation. Because this proxy service is not specific to a single producer, it should be created in the wsrp project folder.

The approach used in this step requires the administrator to assign each producer a name that is included in part of the URL to retrieve the WSDL. The message flow for the proxy service will extract the name from the URL, use it to locate the business service specific to that producer, obtain the WSDL, and then transform the WSDL to rewrite the endpoints to AquaLogic Service Bus. The proxy service endpoint URI is configured as /getWSDL, and the URL that consumers use to obtain a WSDL is:

http://alsb:7001/getWSDL/<producerName>

where <producerName> is the name assigned to the producer by the administrator. In this example, the producer name is operationExample.

Table 3-9 describes how the proxy service is configured:

Table 3-9 Proxy Service Configuration Properties

Property Name	Value	Comments
Service Name	producerWSDL	Any name is allowed.
Service Type	Any XML Service	
Protocol	HTTP	
Endpoint URI	`/getWSDL`	

The message flow for this proxy service consists of a pipeline pair and a route node. The request side of the pipeline pair consists of a single stage whose job is to extract the producer name from the URL and assign it to a context variable. The action is:

```
Assign $inbound/ctx:transport/ctx:request/http:relative-URI to variable
producerName
```

The response side of the message flow is a stage where all the transformations are performed. Before executing the Replace Actions to transform the WSDL, assign the base URL of the AquaLogic Service Bus server to a context variable to avoid specifying it on every transformation:

```
Assign "http://alsb:7001/" to variable nonSecureBaseURL
```

Edit the stage of the Response Pipeline to modify each Replace Action to make the transformation match the Endpoint URI given to the proxies created earlier. In this example, the proxies were created using the producer name with an abbreviated service type appended to it. The `addr` XQuery resource created earlier accepts an extension argument to construct the URI location. Simply change that argument to the proper value, as listed in Table 3-10.

Table 3-10 Extension Settings to Construct the URI Location

If `@binding` is	svc arg of `addr` is
`WSRP_v1_Markup_Binding_SOAP`	`"Base"`
`WSRP_v1_ServiceDescription_Binding_SOAP`	`"Desc"`
`WSRP_v1_PortletManagement_Binding_SOAP`	`"Mgmt"`

Table 3-10 Extension Settings to Construct the URI Location (Continued)

If @binding is	svc arg of addr is
WSRP_v1_Registration_Binding_SOAP	"Reg"
WLP_WSRP_v1_Markup_Ext_Binding_SOAP	"Ext"

You must map "name:" to "$producerName" and "BaseURL" to "$nonSecureBaseURL" similar to the "svg arg" mapping in the use table: table num_xref, "Extension Settings to Construct the URI Location" on page 3-11.

The five Replace Actions are defined as shown in the following code listing. The value of name is replaced with the binding names from the table.

```
Replace
./*[local-name()="definitions"]/*[local-name()="service"]/*[local-name()="
port"][ends-with(attribute::binding,"name")]/*[local-name()="address"
Replace entire node
name
WSRP_v1_Markup_Binding_SOAP
WSRP_v1_ServiceDescription_Binding_SOAP
WSRP_v1_PortletManagement_Binding_SOAP
WSRP_v1_Registration_Binding_SOAP
```

For the first Replace Action, the following User Namespace definitions must be added as in Table 3-11:

Table 3-11 User Namespace Definitions on Replace Action

Prefix	Namespace
wsdl	http://schemas.xmlsoap.org/wsdl/
soap	http://schemas.xmlsoap.org/wsdl/soap/

The route node of this message flow consists of a routing table that selects the case based on $producerName. For each known producer, add cases so that each case routes to the correct business service to retrieve the WSDL if the name matches. This example uses the following directive:

```
= "operationExample" Route to wsdlSvc
```

1. Add a Default Case that routes to the no-op service to handle cases in which an unknown producer name is specified:
   ```
   Default Route to nullSvc
   ```
2. In this example, return an HTTP 404 status code by adding these response actions to the default case:

   ```
   Insert <http:http-response-code>404</http:http-response-code> as last
   child of ./ctx:transport/ctx:response in variable inbound

   Reply With Failure
   ```
3. Edit the Routing Table in the route node to make the cases correspond to the producers known to the system.

Step 5: Verify the Configuration

After completing the configuration, verify it as follows:

1. Retrieve the WSDL from a regular browser window by entering the following URL:

   ```
   http://alsb:7001/getWSDL/operationExample
   ```
2. Verify that all of the end point WSRP end point URLs (except for the BEA extension service) have been changed to correctly refer to the proxy service values on the AquaLogic Service Bus server.
3. Create a remote portlet in a Portal consumer application, specifying this URL as the address of the WSDL for the producer.

   Use either the WebLogic Workshop or Portal Administration Tool to create the remote portlet. Except for entering a different URL to retrieve the WSDL, the steps to create this portlet are the same as those used to create the portlet that is not proxied by AquaLogic Service Bus.
4. After the consumer portal is complete, run the application.
5. Enable monitoring on the AquaLogic Service Bus components that you have chosen.
6. Use the AquaLogic Service Bus Console to drill down to see message counts and performance statistics on all WSRP services and operations handled by the producer.