Gentran Integration Suite™

Web Services

Version 4.3

Sterling Commerce
An IBM Company

Contents

Introduction to Web Services

Web services is a technology that is changing the way of e-commerce. Web services allow us to describe and deploy services in a consistent way so that they can be invoked in a secure and reliable manner. A *web service* is an application that uses these standards and technologies. Web services, software written to link systems over the Internet, are intended to simplify the development of Web-based applications by automating the underlying processes needed for systems to interact online. Web services can be simple or complex. They range from inventory planning tools, mortgage calculators, to a more limited service of looking up a stock quote.

Today, Web services are widely deployed due to use of Extensible Markup Language (XML) and the availability of the Internet. XML is the foundation for the Web Service Description Language (WSDL), which is the standard development tool for Web services in the Application. In addition, XML schemas define the structure and content of the XML documents that are shared. The WSDL contains the following information about the Web service:

- The URL to invoke the Web service: http://*servername*:*SOA_PORT*/soap?service=*WebServiceName*

  where:

  - *servername* is the name of the Application Web Server to be used
  - *SOA_Port* is the port number for this Web service (displayed on last page of Web services group configuration and WSDL generation)
  - *WebServiceName* is the name of the Web service
- The protocols accepted
- Expected input and the data types for each operation
- Output data type for each operation

Note: Currently, Application supports only HTTP and HTTP/s. To use HTTP/s, you must change the HTTP Server adapter configuration to use SSL. For information, see *Services Used By Web Services* on page 43.

Web Services

Application Web services enables you to be both a Web service consumer and a Web service provider. The following are examples of being a Web service consumer and a provider:

- Web services consumer example - By browsing the UDDI directories, you come across a Web service you want to use. You can download the WSDL file that describes it. Next you would need to check in and version the WSDL file in the Application. When the business process goes to use the Web service, the business process queries the checked in WSDL document for connection information. The Web service uses information from WSDL file (description, how to connect to it, and what the service requires and returns) to automate the connection to the Web service.
- Web services provider example - You are the largest customer of a small company. They could run a query against your inventory each night at 2 a.m. and track your inventory levels, which would enable them to plan their production for the upcoming days or weeks more cost effectively. If they see that your inventories levels are dropping faster than usual, they can prepare for a large order from you by putting on more shifts, and increasing production temporarily. If they see that some of your inventory

levels are not moving, they can keep their production in line by running skeleton crews and only one shift—neither you nor the supplier ends up with an overloaded warehouse or inventory write-offs.

What Does Web Services Provide?

Application provides the following Web service functionality:

- Ability to be both Web service consumer and provider
- Ability to create a new Web service
- Ability to check in third party WSDL and use it to create new services (Dynamic Service Creation)
- Ability to view WSDL after it has been generated
- Ability to create input and output schemas and associate them with either a Web service or a business process
- Message reliability, messages are received once and only once
- Security for sending and receiving documents, authentication of users
- WSDL validation for any WSDL checked into the Application

Introduction to Security and Web Services

When making decisions about the security of your Web services, keep the following in mind:

- Encryption is not supported in the Application Web services.
- Requiring signed requests is optional, but strongly suggested. To use signed requests, your consumer will have to obtain a CA certificate and send you the public key. Web services verifies signatures on both messages and attachments.
- Signing outgoing responses is optional, but strongly suggested. To use signed responses, you must generate a system certificate in the Application and send your consumer the public key. Web services signs both messages and attachments.

In addition, prior to configuring your Web service, you will need to:

- Meet with the Web service consumer to decide whether messages and attachments must be signed.
- If requests must be signed, have the consumer obtain a CA certificate and send you the public key.
- If responses must be signed, generate a system certificate to use for responses to the consumer. Send the consumer the public key.
- Check in the certificates to the Application and include the certificates in a Web services configuration.

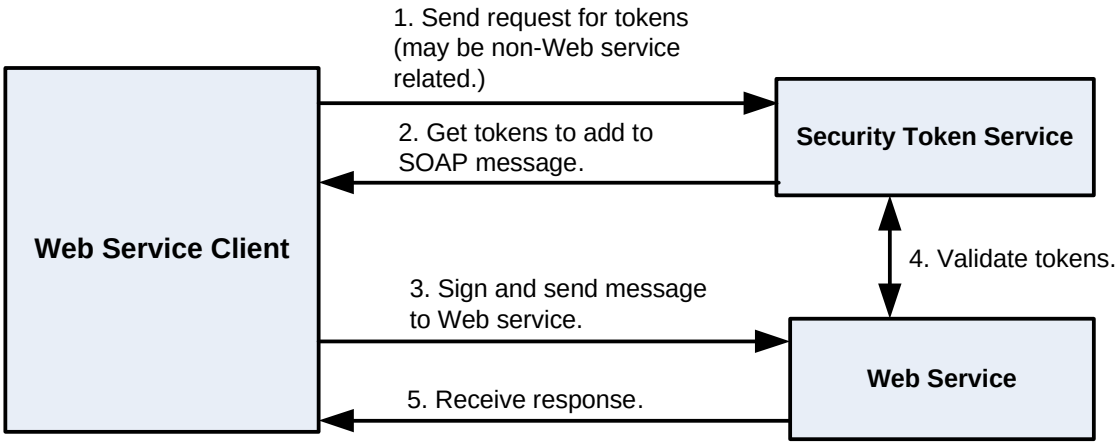
Note: Use a certificate in the Application for only one function at a time. In other words, do not use the same certificate for multiple consumers or for different functions, such as Web service message authentication and EDI.

How is Security Implemented in Web Services?

Application builds upon the existing security technologies, such as XML digital signatures and X.509 certificates, to deliver an industry standard way of securing web services message exchanges. In Application, Web services security features include:

- Authentication of the message sender
- Ensure message integrity
- Prevent replay attacks

The following figure illustrates the flow of a security token in Application:



What is a UserName Token?

Application allows you to create a plain text (UserName) security token to credential and identify yourself when sending and receiving documents using web services. If you are going to use a UserName token in your web service, you need to add it to the Application before you configure the web services group. In addition, you have the option of creating a digested security token. A digested token is more secure as it is encrypted using a nonce (a randomly generated number) and timestamp (date and time).You can also use a UserName token with other applications that support security tokens within the Application. However, security tokens imported or created using the web services security token feature are not listed on the Application certificates page.

What is a X.509 Certificate Token?

When you configure a Web service in the Application, you have the option of specifying the request and response security settings. This option allows you to add or create a certificate token for verification or security purposes. As part of the web services response security settings, if you choose to add a security header and a certificate token, it will automatically be stored as an X.509 signing certificate.

You can also select the way in which the X.509 certificate will be embedded in the security header:

- BinaryToken—Sends the signing X.509 certificate as a BinarySecurityToken.
- IssuerSerial—Sends the issuer name and serial number of a certificate to the receiver. This is the default.
- X509KeyIdentifier—Sends the X.509 certificate used to encrypt the symmetric key.

Message Reliability in Web Services

WS-Reliability is a SOAP-based specification that provides reliable messaging requirements for Web services. Reliable messaging means to deliver a message once and only once to its intended receiver.

The Application reliability feature provides:

- Messages are delivered once and in the correct order.
- Reliable Messages (RM) with large attachments can be processed.
- SOAP 1.1 and SOAP 1.2 message types are supported.
- Request-Response Message Exchange Pattern - The SOAP response is sent back from a SOAP request.
- Response Reliable Message (RM)-Reply Pattern - Responses are only provided for RM types, if any other message type are received (Callback, Poll, etc.), then the RM-Reply Pattern will be a Feature Not Support error message back to the user.
- RM Response element will always be included as a SOAP header in outgoing SOAP responses.
- Quality of Service (QoS) standard is supported. The following table illustrates valid quality of service element combinations:

	In_Order	**Exactly_One**	**At_Most_Once**	**At_Least_Once**
Acknowledge	Yes	Yes	No	Yes
Duplicate	Yes	Yes	Yes	No
Ordered	Yes	No	No	No

  Messages with any other combinations will result in an exception message back to the originator.
- Message configuration options includes:
  - Reliable - only reliable messages are accepted
  - Non-reliable - only non-reliable messages are accepted
  - AutoDetect - both types of messages are accepted

Using Synchronous Mode in Web Services

Currently, HTTP Server Adapter supports bootstrapping a business process in asynchronous mode. The HTTP Server Adapter configurations have been updated with an option to initiate a business process in synchronous mode. In synchronous mode, HTTP Server Adapter executes the business process in the same thread and will wait for processing to be completed.

Since Web service invocation occurs through an HTTP Server Adapter, you can bootstrap a business process in synchronous mode. In order to differentiate the bootstrapping of a business process in synchronous mode, one new URI (/soap-sync) has been added in the existing Web service HTTP Server Adapter configuration.

During Web service configuration, you can decide whether to invoke a Web service end-pt in synchronous mode or not. Based on the selections, the appropriate end-pt address will be generated in the WSDL. When configuring a Web service:

- If you select Use Synchronous Mode option, the generated WSDL will have /soap-sync URL associated with it.

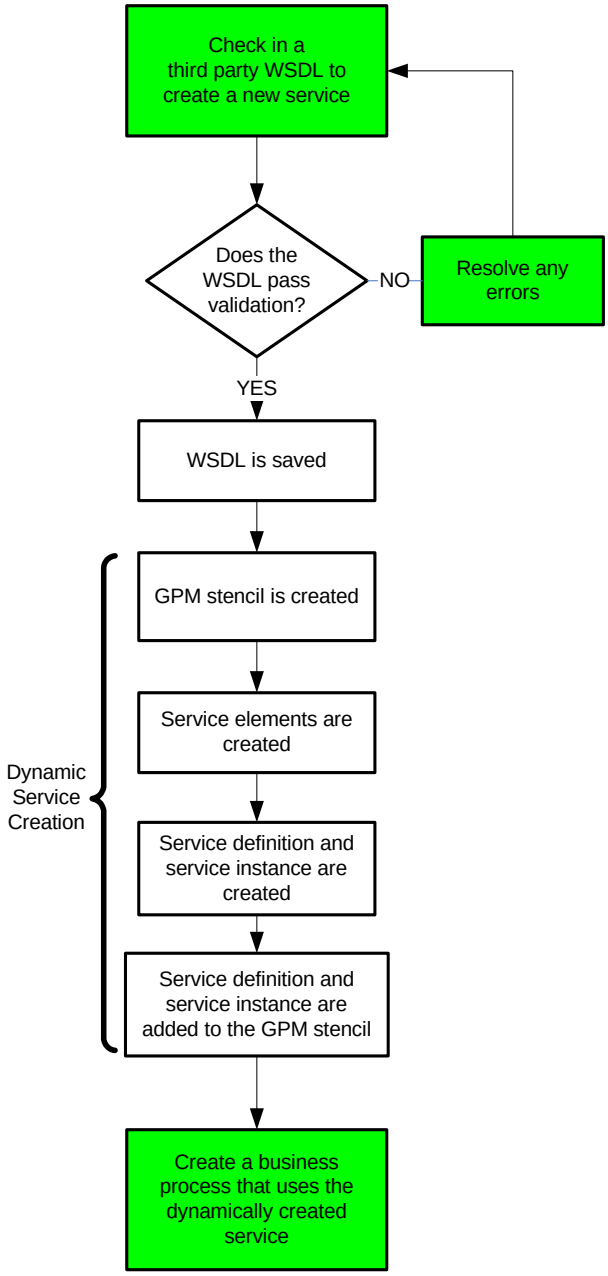
When making deciding to use synchronous mode over asynchronous mode, keep the following in mind:

- Synchronous mode invocation is ideal for a service end-pt compared to a BP end-pt.
- In synchronous mode, avoid sending large attachments to the service end-pt.

Introduction to Dynamic Service Creation

Dynamic Service Creation enables you to check in a third party WSDL, validate the WSDL, and generate all the service definitions and service configurations required for a new service that can be used to access a

third party Web service. You can use these dynamically created services in business processes. The following graphic illustrates the Dynamic Service Creation process:



Note: Shaded shapes are steps the user must do. Shapes in white are internal Gentran Integration Suite activities.

For more information about the Dynamic Service Creation process, see *Dynamic Services*.

XML Schemas in Web Services

An *XML schema* describes the structure of an XML document. A valid XML document must be well formed and must be validated. A schema defines data types. Data types can be anything from simple to complex.

An XML schema defines:

- What elements can appear in the document
- What attributes can appear in the document
- What elements are child elements, sequence in which they appear, and the number of child elements
- Whether an element can be empty
- Default values for attributes

The Application uses 2 types of schemas for Web services, input and output. These input and output schemas are mapped to a business process during Web service configuration. The Application provides a default input and output schema. They contain two parameters: Process Data and Primary Document. These default schemas do not validate the data and do not contain any restrictions on the structure or type of data in the message. They are simply there for use by the Application if no other schema is provided.

The best practice is to create input and output schemas for each business process you want to use in a Web service to ensure that data is structured correctly.

Schema Limitations for Business Processes

The following limitations apply to business process schemas:

- Each business process can have only one root element mapped to it per input or output schema.
- Valid schemas do not need to have a root element. However, if you want to map a business process to a schema, it must have at least one root element.
- You cannot map the same root element to multiple business processes. There should always be a one-to-one mapping relationship between a business process and a root element.
- If a schema/root element combination has already been used with a business process, you cannot use the same combination again, even with a different business process.
- Namespace must be present in the schema for the WSDL to generate properly.

Note: You must add your XML schemas to the Application before you can map them to an existing business process.

Note: The Web services layer will not publish the output schema in the generated WSDL. If the Web Service end point business process has a mapped input and output schema, the generated WSDL will contain only the root-element of the input schema and not the output schema.

Schema Limitations for Web Services

XML schemas for Web services have the following constraints:

- The schemas must contain a targetNamespace.

- The schemas must contain only one root element.

The basic structure must include the following:

Node	Description
Type of data	String, other types accepted. Required. Default is String.
Document encoding type	Required. Default is UTF8. Also accepts UTF16 (this is a SOAP specification).

Note: When creating schemas, give them descriptive names that include information such as the business function or consumer name/type. Include the direction (input or output), if the schema will be used for only one direction.

WSDL Validation

During the WSDL check in process, the Application automatically validates the WSDL file against the following rules:

Note: If any validation criteria is not met by the WSDL, the system generates an error message stating that the WSDL is invalid with an explanation for the error.

- Well-formedness– The WSDL file must be a well-formed XML document. The system will not allow you to check in a text file or non-XML file.
- WSDL Syntax– The Application loads the WSDL definition from the WSDL file and generates an error if basic WSDL syntax is not adhered to by the WSDL file.
- Services in WSDL – The WSDL document should contain at least one service defined in it and each service should have a name associated with it.
- Ports – There should be one or more ports defined in a particular service and each port should have a distinct name associated with it.
- SOAP Ports – There should be at least one port with a SOAP address location specified in the WSDL. Ports with other addresses like the HTTP address and the FTP address are ignored.
- End Point – Each SOAP address should have a location attribute that specifies the End Point where the web service is running. The End Point location should be a proper URL and the protocol specified should be HTTP.
- Operations - The number of operations in the Binding section of the WSDL and the Port Type section of the WSDL should match and their names have to be the same.
- Input/Output Messages- Each operation should have an input and output operation associated with it. The Dynamic Service Creation supports only Request-response transmission and does not support Solicit-response or one-way transmission types.
- Parts – Each message in the WSDL can have zero or more parts associated with it. Each part inside a message element should have a distinct name associated with it.
- Part Element – In case of a document/literal binding style each part should have a part element associated with it that refers to a root element of the schema defined in the WSDL.
- Binding Information – Each port should refer to a binding element within the WSDL using the binding attribute. The binding element gives information about the type of binding to use (RPC/Document) and gives information about the nature of the input and output message elements. The Application dynamic services only supports SOAP binding over the HTTP protocol:
  - When you check in a WSDL that contains only non-SOAP bindings, the WSDL is considered invalid and is not allowed to be checked into the system.
  - When a WSDL contains both SOAP as well as non-SOAP bindings, the non-SOAP bindings are ignored and the dynamic services are only created for the SOAP addresses.
  - When a WSDL contains one SOAP address and two HTTP addresses (corresponding to HTTP Get and HTTP Post bindings), the dynamic services are created only for the port named SendSMSSoap and the other two ports are ignored.
- Duplicate operations:
  - The maximum number of duplicate operations for a WSDL with an RPC style binding is two.

- The WSDL is rejected as invalid if there are three or more operations with the same name inside the same port type element.
- If there are exactly two operations with the same name, the WSDL is checked in successfully into the system. This is a Application known issue and the WSDL is internally parsed.
- Document/Literal WSDL duplicate operation names are never allowed (not even two).

Web Service Configuration via URL

During the Application Web service configuration a base URL is provided for the Web service. The URL is displayed after the Web service configuration has been successfully completed. In addition, when you list all available web services, a link to the web service is provided in the View WSDL field.

Planning for a New Web Service

During the Web service creation process, you are able to locate and configure many components that are already part of the Application. You can use almost any business process, service, or adapter as a Web service component (except internal services and business processes).

Consider the following, as you begin to plan for a new Web service:

Item	What to Consider
Services or adapters	What one or more services or adapters could this new Web service invoke when a request from a consumer is received? Hint: Generally, it is suggested that you create a unique configuration of any service or adapter for use with your Web service.
Business processes	What one or more business processes could this new Web service invoke when a request from a consumer is received? Hint: When preparing to use a business process as part of a Web service, ensure that it is the default version of the business process. Ensure that all services and adapters are enabled.
Input XML Schema	Do I want the Application to use a specific input schema when processing a file from a Web service consumer? Hint: You can associate an input XML schema with a business process in the Web services group.
Output XML Schema	Do I want a customized or default response from the Application? Hint: To generate a customized response based on the output schema, create your own version of the built in SOA_RequestHandler business process and exclude the SOA Response Builder service from it.
Message Reliability Settings	What type of messages will my Web service accept? The Application supports the following reliability settings: ◆ AutoDetect ◆ ReliableOnly ◆ NonReliable Caution: You cannot send the same reliable message (groupID and sequenceNo) to two different web services configured in the same Application instance.
WS-I Conformance Settings	What type of messages conformance will my Web service have? The Application supports the following conformance types: ◆ Conforming WSDL ◆ Conforming SOAP Response
Attachment Settings	What type of attachments will my Web service accept? The Application supports the following types of attachments: ◆ Input ◆ Output ◆ Inline

Item	What to Consider
Certificates	Web services supports message signing for both incoming and outgoing messages. Web services uses the following types of certificates: ◆ Trusted Certificate – Used to verify signatures received from a Web service consumer ◆ System Certificate (X.509)– Used to sign the Application response to a consumer
Consumer Access to the Web Service	In order for a consumer to access and use your Web service, you must create a Application user account for them. The user account must have access rights to all of the components included in the Web services group (services, adapters, business processes).

Other planning activities include:

- ✦ Collect the Web service provider information
- ✦ Collect the Web service consumer information
- ✦ Set up any new User Accounts in the Application

Developing a New Web Service Checklist

Once you have identified a need to provide a new Web service, use this checklist to guide you through the implementation process:

Task	Description
1	Plan the business function on paper.
2	Identify the consumer who requires access to this Web service (your partner). Create a user account for each consumer in the Application and send the user account and password to the consumer.
3	Fill out the Collecting Consumer Information and Collecting Provider Information Checklists. The two checklists are designed to assist you and your consumer to complete the process of developing and implementing a new Web service by keeping relevant information in two documents.
4	Discuss security requirements with your Web service consumer: ◆ Will messages be signed? ◆ What type of certificate will they use (CA or other third party) ◆ When will certificates be obtained? ◆ How will the public keys and passphrases be exchanged (e-mail, phone, letter)?
5	Create a system certificate, if required.
6	Send the consumer the public key for the certificate.
7	Check in the certificate to the Application.
8	Once you have received the public key for your consumer's certificate, check it in to the Application.
9	Decide which Application components (services, adapters, business processes) you need to create for this Web service.
10	Create the necessary service and adapter configurations and business process models. Ensure that the service configurations are enabled. If you want to use HTTP/s instead of HTTP, modify the SOA HTTP Server adapter configuration.
11	Create the necessary business processes, check in new business process models, and enable them. Only the default version of each business process is available for Web services, so if you have more than one version of the business process, ensure that the correct version is the default.
12	Create input and output XML schemas for the business process models in this Web service. (If you do not provide input and output XML schemas, default schemas in the Application are used. The default schemas do not perform any typing or validation).
13	Check in the XML schemas to the Application.
14	Create a Web service in the Application.
15	Map the XML schemas to the appropriate business process models in the Application.

Task	Description
16	Generate the WSDL for the Web services group in the Application. Send the WSDL to your consumers or enable them to access it from a URL on your Application web server. (To publish the Web service to a UDDI, use a third party tool.) **Tip:** Send the consumer your WSDL as soon as possible so that they can determine how to create the messages.
17	Test the Web service.
18	Once testing is complete, notify the consumer that the service is ready for use.

Collecting Provider Information for a New Web Service

Collect the following information when creating a new Web service:

Item	Description
URL for Web service	URL: _____ Sent to consumer on (date): ____/____/____
URL for WSDL	URL: _____ Sent to consumer on (date): ____/____/____
User Account	User ID: _____ Password: _____ Sent to consumer on (date): ____/____/____
File Specifications	Sent to consumer on (date): ____/____/____
XML Schemas	Input XML schema name: _____ Delivered on (date): ____/____/____ Tested on (date): ____/____/____ Output XML schema name: _____ Delivered on (date): ____/____/____ Tested on (date): ____/____/____
If you and your Web service consumer require that messages are signed:	
Certificates to be obtained by the consumer	Public key received from consumer on (date): ____/____/____ Certificate Name/ID: _____
Certificates to be obtained by you (provider)	Public key sent to consumer on (date): ____/____/____ Certificate Name/ID: _____
Testing	WSDL URL tested on (date): ____/____/____ Comments: Web service URL tested on (date): ____/____/____ Comments:

Collecting Consumer Information for a New Web Service

Collect the following information when creating a new Web service:

Item	Description
URL for Web service	URL: _____ Sent to consumer on (date): ____/____/____
URL for WSDL	URL: _____ Sent to consumer on (date): ____/____/____
User Account	User ID: _____ Password: _____ Sent to consumer on (date): ____/____/____
File Specifications	Sent to consumer on (date): ____/____/____
XML Schemas	Input XML schema name: _____ Delivered on (date): ____/____/____ Tested on (date): ____/____/____ Output XML schema name: _____ Delivered on (date): ____/____/____ Tested on (date): ____/____/____
If you and your Web service consumer require that messages are signed:	
Certificates to be obtained by the consumer	Public key received from consumer on (date): ____/____/____ Certificate Name/ID: _____
Certificates to be obtained by you (provider)	Public key sent to consumer on (date): ____/____/____ Certificate Name/ID: _____
Testing	WSDL URL tested on (date): ____/____/____ Comments: Web service URL tested on (date): ____/____/____ Comments:

Creating a New Web Service

To create a new a Web service:

1. From the **Deployment** menu, select **Web Services** > **Manager**.
2. Under Create, next to Create a Web Service Configuration, click **Go!**
3. Complete the fields displayed in the wizard and click **Next** to advance.
4. Confirm your selections and click **Finish**.
5. You have finished this procedure. Click **Return** to continue.

Note: Once the Web service has been created, the following message displays:

The base URL for your Web service is **http://*serverIPaddress*:*00000***

Where *serverIPaddress* is the IP address of the server and *00000* is the port number that you should reference for this Web service.

Note: Sterling Commerce recommends you to use the new Web services request and response settings to take the full support of the WS-Security specifications.

Web Service Name Field Definitions

Field	Description
Name	Unique name for the Web services group. Required.
Description	Description for the Web services group. Required.
Use Synchronous Mode	Check the box if you want to invoke Web services in a synchronous mode.

Web Service Request Security Settings Field Definitions

Field	Description
Verification Certificate	To use a verification certificate (trusted) with the security header, type the certificate name or click the list icon to access available certificates. Select a certificate from the list and click **Save**. Required if Security Header is yes and Username token is not used.
Security Header	Select **Yes** if you want to include a security header in the request. If selected, a verification certificate or UserName token, or both, must also be added. Optional.
UserName Token	To add a UserName token with the security header, check the box and select a UserName token from the list. Required if Security Header is yes and verification certificate is not used.

Web Service Response Security Settings Field Definitions

Field	Description
Signing Certificate	To use a signing certificate (system) with the security header, type the certificate name or click the list icon to access available certificates. Select a certificate from the list and click **Save**. Required if Security Header is yes and Username token is not used.
Security Header	Select **Yes** if you expect a security header in the response. If selected, a signing certificate (X.509) or UserName token, or both, must also be added. Optional.
Refer X.509 Certificate as	If you use a signing certificate (system), it is in X.509 format. Select how the certificate should be embedded in the Security Header. Required if Security Header is Yes. Valid values are: ◆ BinaryToken—Sends the signing X.509 certificate as a BinarySecurityToken. ◆ IssuerSerial—Sends the issuer name and serial number of a certificate to the receiver. This is the default. ◆ X509KeyIdentifier—Sends the X.509 certificate used to encrypt the symmetric key.
UserName Token	To use a Username token with the security header, check the box and select a UserName token from the list. Required if Security Header is yes and signing certificate is not used.

Assign Business Processes Field Definitions

Field	Description
Filter by name	To filter the list of available business processes, type part of the business process name into the Filter field and click the **Filter** icon. Optional.
Available and Selected lists	Select one or more business processes from the list of available business processes on the left to be associated with this Web services group and click the right arrow. To select all available business processes, click the double right arrow. Optional.

Web Service Assign Service Instances Field Definitions

Field	Description
Filter by name	To filter the list of available services, type part of the service name into the Filter field and click the **Filter** icon. Optional.
Available and Selected lists	Select one or more services from the list of available services on the left to be associated with this Web services group and click the right arrow. To select all available services, click the double right arrow. Optional.

Web Service Assign Consumers Field Definitions

Field	Description
Filter by name	To filter the list of available users, type part of the user name into the Filter field and click the **Filter** icon. Optional.
Available and Selected lists	Select one or more users from the list of available users on the left to be associated with this Web services group and click the right arrow. To select all available services, click the double right arrow. Optional.

Note: If you select a consumer, the **mesaAuth** element is inserted into the input message of the generated WSDL. If you do not select a consumer, the **mesaAuth** element is not inserted into the input message of the generated WSDL.

Web Service Reliability Settings Field Definitions

Caution: You cannot send the same reliable message (with the same groupID and sequenceNo) to two different web services configured in the same Application instance, or the operation will fail.

Field	Description
AutoDetect	Both reliable and non-reliable messages are accepted. This is the default. Optional. However selection of AutoDetect, ReliableOnly, or nonReliableOnly is required. **Note:** Select this option when using the Outbound Attachment option to send a single attachment with a SOAP Outbound response.
ReliableOnly	Only reliable messages are accepted. Optional. However, selection of AutoDetect, ReliableOnly, or nonReliableOnly is required.
NonReliableOnly	Only non-reliable messages are accepted. Optional. However, selection of AutoDetect, ReliableOnly, or nonReliableOnly is required.

Web Service WS-I Conformance Settings Field Definitions

Field	Description
Conforming WSDL	Adds a WS-I claim element in every conforming node of WSDL. Optional.
Conforming SOAP Response	Adds a WS-I claim element in the header of a SOAP response. Optional.

Web Service Attachment Settings Field Definitions

Field	Description
Input Attachment	Request can have an attachment. Optional.

Field	Description
Output Attachment	Response can have an attachment. Optional. **Note:** Do not select if sending a SOAP Outbound response containing multiple attachments. See the SOAP Outbound service documentation for information on how to send multiple attachments.
Inline Attachment	No attachments. Information must be contained inline in the message. Optional.

Testing a Web Service

Before making a Web service available to consumers, you should test the Web service using the following criteria:

1. A consumer is able to connect to your Application server.
2. Security associated with the Web service is operating correctly (including certificates).
3. The WSDL correctly describes the services and business processes and input, so that the Application receives and processes the Web service request from consumers correctly.
4. The services and business processes run properly (no halts, no errors).
5. The expected results are received from the Web service (translated document, completed purchase order, etc.) and are correct.
6. The results are in the correct location (for example, you might have a File System adapter put the results in a file in a specified location on a hard disk somewhere), or are sent to the correct recipient.

Deleting a Web Service

Note: Before you delete a Web service, it is recommended that you complete the following tasks:

- Notify consumers that the Web service will no longer be available.
- Use the Export Resources function to save a copy of the Web service to offline storage.

To delete a Web service:

1. From the **Deployment** menu, select **Web Services** > **Manager**.
2. Do one of the following:
   - Type the name of the Web service in the Search field and click **Go!**
   - Select the first letter of the Web service name from List Alphabetically and click **Go!**
   - Click **Go!** next to List Alphabetically.
3. Check the delete checkbox next to the name of the Web service to be deleted and click **Delete Selected Items**.
4. A warning message is displayed. Verify that you want to delete this Web service and click **OK**.
5. A confirmation window is displayed. Verify that the information shown in is for the correct Web service and click **Delete**.

Generating WSDL for a New Web Service

To generate the WSDL for a new a Web service based on the configuration:

1. From the Web Services Manager page, select the Web service by typing the name in the Search By Name field or by selecting the first letter of the name in the alphabetical list. Click **Go!**
2. On the Web services results page, click **Generate WSDL** next to the name of the Web service.
3. A Confirm page displays. Click **Finish** to generate the WSDL.

Note: Once the WSDL is generated, the following message is displayed:

WSDL Generation has been completed successfully.

The base URL for your web service is **http://*serverIPaddress*:*00000***

Where *serverIPaddress* is the IP address of the server and *00000* is the port number that you should reference for this Web service. You should record this information, as you will need this address for your consumers.

4. Do one of the following:
   - Click **Return**.
   - Click **View WSDL**, and then review the WSDL, close the file when completed.
   - Click **Download WSDL**, follow the instructions for downloading the file to your desktop.
5. You have finished this procedure. Click **Return** to continue.

Viewing a WSDL for a Web Service

To view a WSDL for a Web service:

1. To find the link (URL) to the Web service, from the Administration menu, select **Deployment** > **Web Services** > **Manager**. On the Web services manager, click **Go!** next to List All.
2. Click **View WSDL**.

Making the Web Service Available to Your Users

In order for your users to access the Web service, you will need to either distribute or publish the generated WSDL for the Web service:

Do one of the following:

- You can distribute the WSDL as a file to your consumers.
- You can provide a URL to your consumers.

  The URL would be similar to the following example:

  http://*serverIPaddress*:*SOA_PORT*/wsdl?configName=*WebServicesGroupName*

  where:

  - *serverIPaddress* is the IP address of the Application Web Server to be used
  - *SOA_Port* is the port number for this Web service (displayed on last page of Web services group configuration and WSDL generation)
  - *WebServiceName* is the name of the Web service.

Note: The Application will not automatically publish WSDL to a UDDI. To do so, you must use a third party product.

Checking In WSDL for a Web Service

To check in a WSDL:

1. From the **Deployment** menu, select **Web Services** > **WSDL Check In**.
2. Click **Go!** next to Check In New WSDL.
3. Complete the fields displayed in the wizard and click **Next** to advance.
4. A Confirm page displays. Click **Finish** to check in the WSDL.
5. You have finished this procedure. Click **Return** to continue.

Naming Field Definitions

Field	Description
Name	A unique name for the WSDL. Required.
Select an input mode for defining the new WSDL	Required. There are two ways to check in a new WSDL file: ◆ Check in WSDL – Use to import a WSDL file. ◆ WSDL Text Editor – Use to write a new WSDL or paste from a text editor or other program.

Naming - Select WSDL Field Definitions

Field	Description
WSDL filename(.wsdl)	Enter the name or browse and select the WSDL from your system. Required.
Check-in Comments	Enter any comments necessary. For example, enter a version number or whether this is for test or production. Required.
Encoding Type	Select from the list. Default is UTF-8.

WSDL – Confirm Field Definitions

Field	Description
Enable for Business Processes	Leave selected to enable this WSDL for use with business processes.

If you selected WSDL Text Editor, the following fields are displayed:

Naming – Edit WSDL Field Definitions

Field	Description
Description	A description for this WSDL. Required.
WSDL	Type the WSDL or paste it from a text editor or other program.

Checking In a New Version of WSDL

To check in a new version for a WSDL:

1. From the **Deployment** menu, select **Web Services** > **WSDL Check In**.
2. In the List section, select **ALL**, and click **Go!**
3. Next to the WSDL file for which you want to check in a new version, click **source manager**.
4. Click **Go!** next to **Check in a new version of this WSDL**.

   You can also click **check in** next to a WSDL to disable the lock and check in a WSDL.
5. Select the WSDL file document (.wsdl file) from the directory on your client computer where you saved it.
6. Type check in comments that help identify this version of the WSDL.
7. Select an encoding type. Click **Next**.
8. To make this WSDL the default version, select the WSDL under **OTHER Versions**.
9. To enable the WSDL for business processes, click **Next** and verify that the check box next to **Enable for Business Processes** is selected.

   If you choose not to enable this WSDL for business processes, clear the **Enable for Business Processes** check box.
10. To unlock the WSDL file, verify that the check box next to **Release the lock on the file** is selected.
11. Click **Finish**.

Deleting a WSDL

Note: Before you delete a WSDL file, use the Export Resources function to save a copy of the WSDL to offline storage.

To delete a WSDL file:

1. From the **Deployment** menu, select **Web Services** > **Manager**.
2. Type the name of the WSDL file in the Search field and click **Go!** or select the first letter of the WSDL file name from List Alphabetically and click **Go!**

   All WSDL files for your selection display.
3. Click the name of the WSDL file to be deleted. (The WSDL file settings display in a separate window.)
4. In the new window, verify that this is the WSDL file to delete and click **Close**.
5. Select **Delete** next to the WSDL file that you want to delete and click **Delete Selected Items**.
6. A warning message is displayed. Verify that you want to delete this WSDL file and click **OK**.
7. A confirmation window displays. Verify that the information shown is for the correct WSDL file and click **Delete**.

Creating an XML Schema for Web Services

To create a input and output XML schemas for Web services:

1. Create the input and output XML schemas for your business process using an XML text editor.

   Note: You can use only XML schemas (.xsd) with Web services; you cannot use DTD (.dtd) files.
2. Check in the XML schemas to the Application.
3. Map the XML schemas to their business process in the Application.

Mapping XML Schemas to Business Processes

To map an XML schema to a business process:

1. In Application, select **Deployment** > Web Services > **Schema Mappings**.
2. Under Create, next to **Create a New BP Schema Mapping**, click **Go!**
3. Complete the fields displayed in the wizard and click **Next** to advance.
4. Confirm that the choices displayed are correct and click **Finish** to add the BP schema mapping.
5. You have finished this procedure. Click **Return** to continue.

BP Schema Mapping Field Definitions

Field	Description
Business Process	If this is a new mapping, select a business process to associate input and/or output schemas with. Required. If editing an existing relationship, this field is unavailable.
Input Schema	Select an XML schema to be used as the input schema for the business process or service. By default, Web services layer does not validate the input data to a business process against the input schema. It just passes the received data to the business process or service. **Note:** An Input Schema or an Output Schema must be specified. If you select only an output schema, the system will use the generic input XML Schema with it.
Output Schema	Select an XML schema to be used as the output schema for the business process or service. By default, Web services layer does not format the response based on the output schema before sending the response to the sender. **Note:** An Input Schema or an Output Schema must be specified. If you select only an input schema, the system will use the generic output XML Schema with it.

BP Root Element Mapping Field Definitions

Field	Description
Input Root Element	Select a Root Element for the Input Schema to define the input formats for a business process. Required if Input Schema previously specified.
Output Root Element	Select a Root Element for the Output Schema to define the output formats for a business process. Required if Output Schema previously specified.

Deleting a BP Schema Mapping

To delete a BP Schema Mapping:

1. On the BP Schema Mapping page, do one of the following:
   - Type the name of the business process in the Search field and click **Go!**
   - Select **ALL** or the first letter in the name of the business process from List Alphabetically and click **Go!**
2. Select the name of the business process. The BP Mapping configuration displays in a new window. Review the input and output schemas and verify that this is the correct BP Schema mapping to delete. Click **Close** to close the window.
3. Click **Delete** next to the name of the business process.
4. The message, “Are you sure you want to delete this BP Schema Mapping?” displays. Click **Yes**.
5. A confirmation page displays. Click **Delete** to confirm and delete this BP Schema mapping.
6. You have finished this procedure. Click **Return** to continue.

Creating a UserName Security Token

To create a UserName security token:

1. From the Administration menu, select **Deployment** > **Web Services** > **Security Tokens**.
2. Under Create, next to Create Security Token, click **Go!**
3. Complete the fields displayed in the wizard and click **Next** to advance.
4. Confirm that the choices displayed are correct and click **Finish**.

Security Token Field Definitions

Field	Description
Security Token Name	Type a name for this token. Required.
Token Description	Type a description for this token. Required.
Token Type	Select a token type. Required.

UserName Token Field Definitions

Field	Description
User Name	Type a user name to be used with this token. Required.
Password	Type a password to be used with this token. Required.
Digest	Select if you also want the password to be digested instead of being stored as plain text. Optional.

Deleting a Security Token

To delete a security token:

1. From the Administration menu, select **Deployment** > **Web Services** > **Security Tokens**.
2. Do one of the following:
   - Under Search, next to Security Token Name, type the name of the token you want to edit, then click **Go!**
   - Under List, select alphabetical or token type search parameters, then click **Go!**
3. Next to the token name you want to delete, click **source manager**.
4. Do one of the following:
   - Under Delete, select one or more of the token versions you want to delete and click **Go!** next to Delete Selected Versions.
   - Next to Delete All versions, click **Go!**
5. Confirm your delete selection and click **OK**.
6. On the Resource Summary page, click **Next**.
7. Click **Delete**.

Editing a Security Token

To edit a security token:

1. From the Administration menu, select **Deployment** > **Web Services** > **Security Tokens**.
2. Do one of the following:
   - Under Search, next to Security Token Name, type the name of the token you want to edit, then click **Go!**
   - Under List, select alphabetical or token type search parameters, then click **Go!**
3. Next to the token name you want to edit, click **source manager**.
4. Next to the version you want to edit, click **edit**.
5. Make changes as needed to the token information and click **Next**.
6. On the Security Token Version page, select the version you want as the default and click **Next**.
7. Click **Confirm**.

Services Used By Web Services

The following table contains the names and descriptions of the Application services and adapters used to enable Web services. These services are internal and should not be changed or copied.

Service	Description
SOA Http Server Adapter	Configuration of the HTTP Server adapter specifically for Web services. As delivered, it is set up to use the default perimeter server (node1 & local) and HTTP. **Note:** This adapter should not be changed or copied, with one exception: You can change the adapter to use HTTP/s by editing the adapter configuration: on the HTTP Connection Properties page of the adapter configuration, select **Must** for Use SSL. Click **Save**, then **Finish**.
Dynamic Service Invoker Service	Invokes a service designated by the service parameter 'SVC_NAME'. The primary business usage of this service will be for Web service invocation. Business process writers can also use this to execute services dynamically in a business process by simply adding the Dynamic Service Invoker (DSI) service as an operation or participant and setting the SVC_NAME parameter.
Service Information Service	Provides information about a referenced service.
SOA Fault Service	Puts the Application specific error information into standard Web services (SOAP) faults. The SOAP protocol requires specific fault handling mechanisms that allow application specific details to be added to the fault message. This service allows the Application detailed error information to be added to the appropriate SOAP fault message.
SOA Inbound	Processes inbound Web service request documents, separates the MIME parts, and validates signatures. Called by the system for a Application Web services consumer process.
SOA Inbound Msg Processing Service	Processes incoming SOAP message. If the incoming SOAP message has MIME attachments, it extracts the attachment parts from the message and uploads the attachments in the process data. Also, it extracts the SOAP Envelope from the message and uploads it in the process data as a primary document. **Note:** Use this service if you don't want any security.
SOA Outbound	Processes the output of the SOA Response Builder service and packages it in MIME format. This service sends an outbound (response) message from a Web service.
SOA Outbound Msg Processing Service	Constructs the final SOAP Response (MIME or without MIME) using the output of the end-point invocation. This service is the complementary to the SOA Inbound Msg Processing Service. **Note:** Use this service if you don't want any security.
SOA Request Handler Service	Processes the output of the SOA Inbound service and prepares the workflow context for invocation. It checks that the request is valid according to the WSDL and that the user is authorized to execute the requested Web service.
SOA Response Builder Service	Processes the output of the invoked Web service and prepares the documents for the SOA Outbound service.

Service	Description
SOA WS Config Info Service	Based on the WSDL service name provided, this service provides information about the referenced service configuration.
HTTP Respond Service	This service is used as part of a business process to return a response to an HTTP request. Used in conjunction with an HTTP Server adapter.
RM Decision Service	Processes inbound Web Services request documents to determine whether the request is a WS-Reliable Message or not.
RM Service Handler	Validates an incoming RM SOAP request message against WS-Reliability_1.1 specification, converts the incoming SOAP message in an internal Message object and then persists this internal Message object into the Workflow Context.
RM Service Manager	Processes, manages, and generates appropriate responses to a WS-Reliable message.

Configuration Overrides for Web Services

Some Web services consumer implementations do not allow use of the SOAP with attachments message format. As a solution, you can override values of some properties in the file soa.properties.in. This file contains several override values that allows you to control the generation of WSDL and the operation of the service provider.

The following list of properties contains descriptions for each of the values that should only be changed when instructed to do so by Sterling Commerce Customer Support, or by expert Application users:

Property	Description
useCache=true	Indicates that generated WSDL will be cached by the Application to speed up retrieval.
class.wsdl=com.sterlingcomerce.woodstock.services.soa.util.WSDLServiceInfo	Class used by Application Service Info service to generate WSDL. **CAUTION:** Do not change this property.
class.xml=com.sterlingcomerce.woodstock.services.soa.util.XMLServiceInfo	Class used by Application Service Info service to generate raw XML representation of services. **CAUTION:** Do not change this property.
wsdlMessage=/wsdl:definitions/wsdl:message	XPath entry of the message element within a WSDL which will required during insertion of the <wsi:Claim> element within the message element. The present of this element indicates the WS-I conformity of the message node in the generated WSDL. **CAUTION:** Do not change the value of this property. Removal of this property indicates that the message node in the Application generated WSDL is not WS-I compliance.
wsdlportType=/wsdl:definitions/wsdl:portType	XPath entry of the portType element within a WSDL which will required during insertion of the <wsi:Claim> element within the portType element. The present of this element indicates the WS-I conformity of the portType node in the generated WSDL. **CAUTION:** Do not change the value of this property. Removal of this property indicates that the portType node in the Application generated WSDL is not WS-I compliance.

Property	Description
wsdlBinding=/wsdl:definitions/wsdl:binding	The XPath entry of the binding element within a WSDL which will required during insertion of the <wsi:Claim> element within the binding element. The present of this element indicates the WS-I conformity of the binding node in the generated WSDL. **CAUTION:** Do not change the value of this property. Removal of this property indicates that the binding node in the Application generated WSDL is not WS-I compliance.
wsdlOperation=/wsdl:definitions/wsdl:portType/wsdl:operation	The XPath entry of the operation element within a WSDL which will required during insertion of the <wsi:Claim> element within the operation element. The present of this element indicates the WS-I conformity of the operation node in the generated WSDL. **CAUTION:** Do not change the value of this property. Removal of this property indicates that the operation node in the Application generated WSDL is not WS-I compliance.
defaultBaseURL=http://&HOST_NAME;:&SOA_PORT;	Default base URL for accessing Web services. Additional HTTP Server adapters can be configured. **CAUTION:** Do not change this property.
defaultSoapURL=http://&HOST_NAME;:&SOA_PORT;/soap	Default SOAP URL for accessing Web services in asynchronous mode. Additional HTTP Server adapters can be configured. **CAUTION:** Do not change this property.
syncBPSOAPURL=http://&HOST_ADDR;:&SOA_PORT;/soap-sync	SOAP URL for accessing Web services in synchronous mode. Additional HTTP Server adapters can be configured. **CAUTION:** Do not change this property.
defaultSOAPort=&SOA_PORT;	Default SOA port for accessing Web services. Additional HTTP Server adapters can be configured. **CAUTION:** Do not change this property.
attachmentMimeType=application/octetstream	MIME type used for SOAP with attachments.
signatureRequired=false	Override to force signature usage.
signatureTrigger=/xmldsig	Indicator used to determine if a given message is signed. **CAUTION:** Do not change this property.
signatureMaxScan=8192	Tuning parameter for signature determination. **CAUTION:** Do not change this property.

Property	Description
enforceStrongTyping=false	When false, all parameters are made optional. This is used when internal service definition is inconsistent with actual usage. Prevents marking of multiple required fields as required.
In addition to the previous parameters, the following are overrides that enable you to tailor the WSDL. These parameters operate at a Web services configuration level which allows for more flexibility: **CAUTION:** Do not change these parameters. These parameters are dynamically populated based on the Web Service Configuration. Modifying these parameters can change the behavior of the configured Web Service.	
wsconfigname.inputHasAttachment=true	When set to false, the generated WSDL will omit the attachment part for the input message. The service provider will not expect an attachment. This can be used when the type parameters are sufficient for operation.
wsconfigname.outputHasAttachment=true	When set to false, the generated WSDL will omit the attachment part for the output message. The service provider will generate responses that contain only a SOAP part.
Wsconfigname.useInlineAttachment=false	When set to true, the generated WSDL will replace the attachment element with an inlineAttachment element, and the binding will be pure SOAP instead of mime/multipart related. Any attached document will be encoded and embedded in the SOAP message itself. This mode is useful when a consumer does not support the SOAP with attachments standard.

SOA Properties Files in Application

There are two soa.properties files - soa.properties.in, which is the template properties file; and soa.properties, which is the live properties file. These are located in your Application /*install_dir*/properties directory.

It is extremely important to ensure that you add the records to the template file, soa.properties.in, not to the live file.

Each time you run the setupfiles command in Application, all live files are updated with the information contained in their template (.in) files. This means that if you make changes to the live file, soa.properties, they are lost each time setupfiles is run. Always make changes to the template file, soa.properties.in, and your changes will be maintained.

To edit the soa.properties.in file:

1. Locate the file in the Application /*install_dir*/properties directory and make changes as required using a text editor for your operating system.
2. After you have completed the edits, run the setupfiles.sh (UNIX) or setupfiles.cmd (Windows) utility located in your Application /*install_dir*/bin directory.

SOAP Fault Messages

The following table includes the SOAP Fault messages used in Web services:

Fault	Description
VersionMismatch	Invalid namespace, local name, or both for the SOAP envelope element.
MustUnderstand	When set to mandatory, indicates that an immediate child element of the Header element was not understood.
DataEncodingUnknown	A SOAP header block or SOAP body child element information item contains a data encoding that the faulting node does not support.
Sender	The message was incorrectly formed or contained incorrect information. The message should not be resent until the information is corrected.
Receiver	There was a problem with the server that prevented the message from proceeding. The message could succeed if resent at a later point in time .

Example of a Provider WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="ExpenseReportDemo"
    targetNamespace="http://www.sterlingcommerce.com/mesa"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:mesa="http://www.sterlingcommerce.com/mesa"
    xmlns:mesa_xsd="http://www.sterlingcommerce.com/mesa/schema"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns0="http://www.sterlingcommerce.com/schema/example/expensereportin"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <wsdl:types>
        <xs:schema attributeFormDefault="unqualified"
            elementFormDefault="qualified"
            targetNamespace="http://www.sterlingcommerce.com/mesa/schema"
            xmlns="http://www.sterlingcommerce.com/mesa/schema"
            xmlns:tns="http://www.sterlingcommerce.com/mesa/schema"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
            <xs:complexType name="Binary">
                <xs:simpleContent>
                    <xs:extension base="xs:base64Binary">
                        <xs:attribute name="href" type="xs:anyURI"/>
                    </xs:extension>
                </xs:simpleContent>
            </xs:complexType>
            <xs:element name="attachment" type="tns:Binary"/>
            <xs:complexType name="ProcessData">
                <xs:sequence>
                    <xs:any/>
                </xs:sequence>
            </xs:complexType>
            <xs:element name="mesaFault" type="tns:MESAFault"/>
            <xs:complexType name="MESAFault">
                <xs:sequence>
                    <xs:element name="code"/>
                    <xs:element name="message"/>
                    <xs:element name="statusReport"/>
                </xs:sequence>
            </xs:complexType>
            <xs:element name="processData" type="tns:ProcessData"/>
            <xs:element name="documents">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element maxOccurs="unbounded" ref="tns:attachment"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:simpleType name="HashType">
                <xs:restriction base="xs:string">
                    <xs:enumeration value="MD5"/>
                    <xs:enumeration value="NONE"/>
                </xs:restriction>
```

```
            </xs:simpleType>
            <xs:complexType name="MESAAuth">
                <xs:sequence>
                    <xs:element name="principal"/>
                    <xs:element name="auth">
                        <xs:complexType>
                            <xs:attribute name="hashType"
                                type="HashType" use="required"/>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="MESAHeader">
                <xs:all>
                    <xs:element name="mesaAuth" type="tns:MESAAuth"/>
                </xs:all>
            </xs:complexType>
            <xs:element name="mesaHeader" type="tns:MESAHeader"/>
            <xs:element name="mesaAuth" type="tns:MESAAuth"/>
        </xs:schema>
        <xs:schema elementFormDefault="qualified"

targetNamespace="http://www.sterlingcommerce.com/schema/example/expensereportin"
            xmlns="http://www.sterlingcommerce.com/schema/example/expensereportin"

xmlns:tns0="http://www.sterlingcommerce.com/schema/example/expensereportin"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
            <xs:element name="ExpenseReportInput">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="EmployeeInformation"/>
                        <xs:element maxOccurs="unbounded"
ref="ExpenseItemInformation"/>
                        <xs:element ref="ExpenseTotals"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="EmployeeInformation">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="Name"/>
                        <xs:element ref="EmpNumber"/>
                        <xs:element ref="SSN"/>
                        <xs:element minOccurs="0" ref="Position"/>
                        <xs:element minOccurs="0" ref="Department"/>
                        <xs:element ref="Manager"/>
                        <xs:element ref="PayPeriodFrom"/>
                        <xs:element ref="PayPeriodTo"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="ExpenseItemInformation">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="Date"/>
                        <xs:element ref="Account"/>
```

```
                    <xs:element ref="Description"/>
                    <xs:element minOccurs="0" ref="Lodging"/>
                    <xs:element minOccurs="0" ref="Transportation"/>
                    <xs:element minOccurs="0" ref="Fuel"/>
                    <xs:element minOccurs="0" ref="Meals"/>
                    <xs:element minOccurs="0" ref="Phone"/>
                    <xs:element minOccurs="0" ref="Entertainment"/>
                    <xs:element minOccurs="0" ref="Other"/>
                    <xs:element ref="Total"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="ExpenseTotals">
            <xs:complexType>
                <xs:sequence>
                    <xs:element minOccurs="0" ref="Lodging"/>
                    <xs:element minOccurs="0" ref="Transportation"/>
                    <xs:element minOccurs="0" ref="Fuel"/>
                    <xs:element minOccurs="0" ref="Meals"/>
                    <xs:element minOccurs="0" ref="Phone"/>
                    <xs:element minOccurs="0" ref="Entertainment"/>
                    <xs:element minOccurs="0" ref="Other"/>
                    <xs:element ref="SubTotal"/>
                    <xs:element minOccurs="0" ref="Advances"/>
                    <xs:element ref="GrandTotal"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="Account" type="xs:string"/>
        <xs:element name="Advances" type="xs:decimal"/>
        <xs:element name="Date" type="xs:date"/>
        <xs:element name="Department" type="xs:string"/>
        <xs:element name="Description" type="xs:string"/>
        <xs:element name="EmpNumber" type="xs:integer"/>
        <xs:element name="Entertainment" type="xs:decimal"/>
        <xs:element name="Fuel" type="xs:decimal"/>
        <xs:element name="GrandTotal" type="xs:decimal"/>
        <xs:element name="Lodging" type="xs:decimal"/>
        <xs:element name="Manager" type="xs:string"/>
        <xs:element name="Meals" type="xs:decimal"/>
        <xs:element name="Name" type="xs:string"/>
        <xs:element name="Other" type="xs:decimal"/>
        <xs:element name="PayPeriodFrom" type="xs:date"/>
        <xs:element name="PayPeriodTo" type="xs:date"/>
        <xs:element name="Phone" type="xs:decimal"/>
        <xs:element name="Position" nillable="true" type="xs:string"/>
        <xs:element name="SSN" type="xs:string"/>
        <xs:element name="SubTotal" type="xs:decimal"/>
        <xs:element name="Total" type="xs:decimal"/>
        <xs:element name="Transportation" type="xs:decimal"/>
    </xs:schema>
</wsdl:types>
<wsdl:message name="MESAResponse">
    <wsdl:part element="mesa_xsd:processData" name="parameters"/>
    <wsdl:part element="mesa_xsd:attachment" name="attachment"/>
</wsdl:message>
```

```
    <wsdl:message name="ExpenseReportDemo">
        <wsdl:part element="mesa_xsd:mesaAuth" name="header"/>
        <wsdl:part element="tns0:ExpenseReportInput" name="parameters"/>
        <wsdl:part element="mesa_xsd:attachment" name="attachment"/>
    </wsdl:message>
    <wsdl:message name="MESAFault">
        <wsdl:part element="mesa_xsd:mesaFault" name="parameters"/>
        <wsdl:part element="mesa_xsd:attachment" name="attachment"/>
    </wsdl:message>
    <wsdl:message name="GISGeneric">
        <wsdl:part element="mesa_xsd:mesaAuth" name="header"/>
        <wsdl:part element="mesa_xsd:processData" name="parameters"/>
        <wsdl:part element="mesa_xsd:attachment" name="attachment"/>
    </wsdl:message>
    <wsdl:portType name="GISPortType">
        <wsdl:operation name="executeExpenseReportDemo">
            <wsdl:input message="mesa:ExpenseReportDemo"/>
            <wsdl:output message="mesa:GISGeneric"/>
            <wsdl:fault message="mesa:MESAFault"/>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="GISBinding" type="mesa:GISPortType">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="executeExpenseReportDemo">
            <soap:operation soapAction="sii:ExpenseReportDemo"/>
            <wsdl:input>
                <mime:multipartRelated>
                    <mime:part>
                        <soap:body parts="header parameters" use="literal"/>
                    </mime:part>
                    <mime:part>
                        <mime:content part="attachment"
type="application/octetstream"/>
                    </mime:part>
                </mime:multipartRelated>
            </wsdl:input>
            <wsdl:output>
                <mime:multipartRelated>
                    <mime:part>
                        <soap:body parts="parameters" use="literal"/>
                    </mime:part>
                    <mime:part>
                        <mime:content part="attachment"
type="application/octetstream"/>
                    </mime:part>
                </mime:multipartRelated>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="ExpenseReportDemo">
        <wsdl:port binding="mesa:GISBinding" name="GISPort">
            <soap:address location="10.11.20.34?service=ExpenseReportDemo"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

Note: The inclusion of the mesaAuth element is optional. If you select a consumer, the mesaAuth element will be inserted into the input message of the generated WSDL and if you do not select a consumer, the mesaAuth element will not be inserted into the input message of the generated WSDL.The inclusion of the mesaAuth element will make the generated WSDL non-compliant with the WS-I Basic Profile 1.1.

Examples of Input and Output XML Schemas for Web Services

Input Schema Example - ExpenseReportInput.xsd

The following is an example of an XML input schema:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="ExpenseReportInput">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="EmployeeInformation"/>
        <xs:element ref="ExpenseItemInformation" maxOccurs="unbounded"/>
        <xs:element ref="ExpenseTotals"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="EmployeeInformation">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Name"/>
        <xs:element ref="EmpNumber"/>
        <xs:element ref="SSN"/>
        <xs:element ref="Position" minOccurs="0"/>
        <xs:element ref="Department" minOccurs="0"/>
        <xs:element ref="Manager"/>
        <xs:element ref="PayPeriodFrom"/>
        <xs:element ref="PayPeriodTo"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="ExpenseItemInformation">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Date"/>
        <xs:element ref="Account"/>
        <xs:element ref="Description"/>
        <xs:element ref="Lodging" minOccurs="0"/>
        <xs:element ref="Transportation" minOccurs="0"/>
        <xs:element ref="Fuel" minOccurs="0"/>
        <xs:element ref="Meals" minOccurs="0"/>
        <xs:element ref="Phone" minOccurs="0"/>
        <xs:element ref="Entertainment" minOccurs="0"/>
        <xs:element ref="Other" minOccurs="0"/>
        <xs:element ref="Total"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="ExpenseTotals">
    <xs:complexType>
      <xs:sequence>
```

```
        <xs:element ref="Lodging" minOccurs="0"/>
        <xs:element ref="Transportation" minOccurs="0"/>
        <xs:element ref="Fuel" minOccurs="0"/>
        <xs:element ref="Meals" minOccurs="0"/>
        <xs:element ref="Phone" minOccurs="0"/>
        <xs:element ref="Entertainment" minOccurs="0"/>
        <xs:element ref="Other" minOccurs="0"/>
        <xs:element ref="SubTotal"/>
        <xs:element ref="Advances" minOccurs="0"/>
        <xs:element ref="GrandTotal"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Account" type="xs:string"/>
  <xs:element name="Advances" type="xs:decimal"/>
  <xs:element name="Date" type="xs:date"/>
  <xs:element name="Department" type="xs:string"/>
  <xs:element name="Description" type="xs:string"/>
  <xs:element name="EmpNumber" type="xs:integer"/>
  <xs:element name="Entertainment" type="xs:decimal"/>
  <xs:element name="Fuel" type="xs:decimal"/>
  <xs:element name="GrandTotal" type="xs:decimal"/>
  <xs:element name="Lodging" type="xs:decimal"/>
  <xs:element name="Manager" type="xs:string"/>
  <xs:element name="Meals" type="xs:decimal"/>
  <xs:element name="Name" type="xs:string"/>
  <xs:element name="Other" type="xs:decimal"/>
  <xs:element name="PayPeriodFrom" type="xs:date"/>
  <xs:element name="PayPeriodTo" type="xs:date"/>
  <xs:element name="Phone" type="xs:decimal"/>
  <xs:element name="Position" type="xs:string" nillable="true"/>
  <xs:element name="SSN" type="xs:string"/>
  <xs:element name="SubTotal" type="xs:decimal"/>
  <xs:element name="Total" type="xs:decimal"/>
  <xs:element name="Transportation" type="xs:decimal"/>
</xs:schema>
```

Output Schema Example - ExpenseReportOutput.xsd

The following is an example of an XML output schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="ExpenseReportOutput">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Approved"/>
        <xs:element ref="ApprovedBy" minOccurs="0"/>
        <xs:element ref="ErrorMsg" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Approved" type="xs:boolean"/>
```

```
  <xs:element name="ApprovedBy" type="xs:string"/>
  <xs:element name="ErrorMsg" type="xs:string"/>
</xs:schema>
```

Web Service Provider Example 1 – Enabling Visible Business

Company A makes earrings. One of their suppliers, Company B, provides the glass beads that are used in the earrings.

Company A has decided that the most efficient way to do business with Company B is to have them check inventory levels through a Web service every night. Company A wants Company B to have real time access to their inventory levels and orders, so that Company B can plan their factory work more efficiently (how many shifts to schedule for the next two weeks, for example). Company A would like Company B to automatically ship them new stock as inventory levels reach certain points. By using Web services, both companies will be able to plan their production and ordering for the upcoming weeks more efficiently.

Company A creates a Web services group called “Inventory_Check.” The Web services group includes the following:

Component	Name
Business Process	Bead_Inquiry.bp
Service	Lightweight JDBC adapter configuration named “Bead_LTWTJDBC” (used in the business process)
User Account	One new Application user account was created: Bead_User

The following activities happen between Company A and Company B:

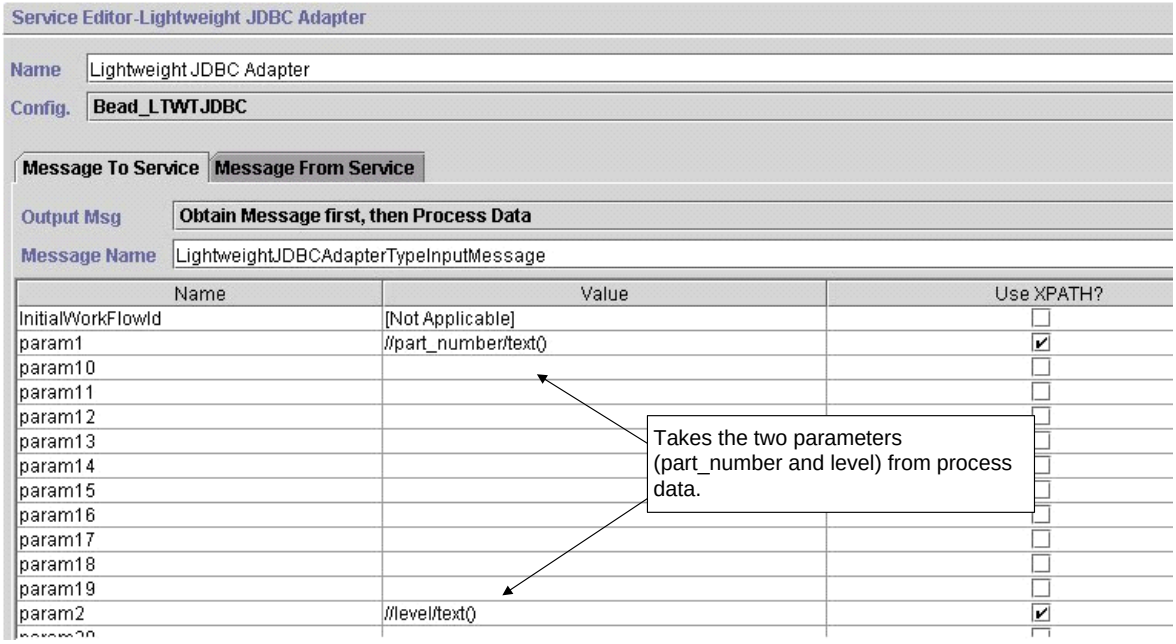
1. Company A creates each of the components listed in the previous table.
2. Company A creates the Web services group, Inventory_Check, then adds the business process and the user account to it.

   Note: It is not necessary to add the Lightweight JDBC adapter configuration, Bead_LTWTJDBC, to the Web services group because it is part of the business process that is being added to the Web services group.
3. Company A generates the WSDL for the Web services group and sends it to Company B (or sends them the URL for the WSDL).
4. Company B accesses the WSDL URL and uses the WSDL as information when creating the query that will be sent to the Web service.
5. Company B creates the query that will be automatically sent each night at 2 a.m. As part of the query, they will include values for two fields in the database, part_number and level.
6. When the query is send to the Web service URL, the Web services invokes the Bead_Inquiry business process. The business process runs the query against the database and returns the response to the Web service. The Web service returns the response to Company B.

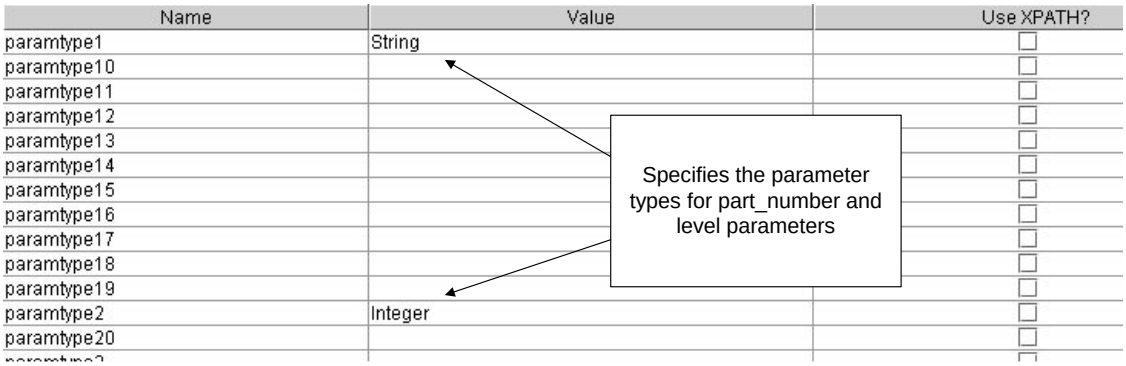
Bead_Inquiry Business Process

The following illustration shows how the business process might look in the GPM:

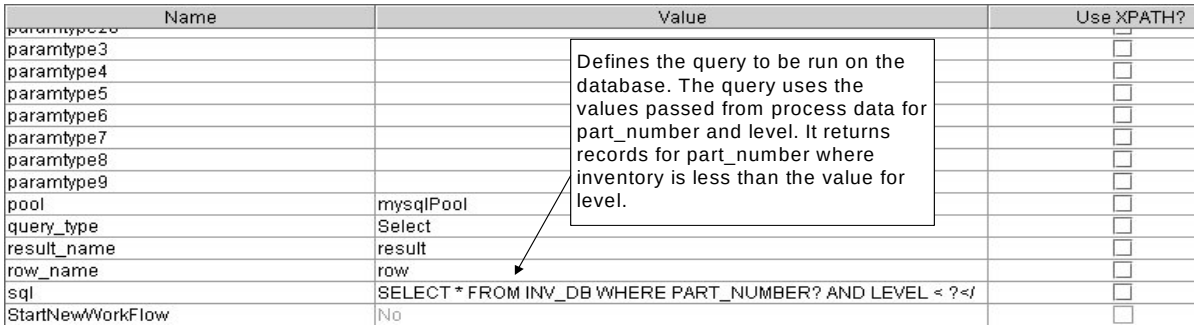
GPM Parameters, Part 1



GPM Parameters, Part 2



GPM Parameters, Part 3



The following BPML is the source for the preceding GPM example:

```
<process name = "Bead_inquiry">
  <sequence>
    <operation name="Lightweight JDBC Adapter">
      <participant name="Bead_LTWTJDBC"/>
      <output message="LightweightJDBCAdapterTypeInputMessage">
        <assign to="param1">//part_number/text()</assign>
        <assign to="param2">//level/text()</assign>
        <assign to="paramtype1">String</assign>
        <assign to="paramtype2">Integer</assign>
        <assign to="pool">mysqlPool</assign>
        <assign to="query_type">SELECT</assign>
        <assign to="result_name">result</assign>
        <assign to="row_name">row</assign>
        <assign to="sql">SELECT * FROM INV_DB WHERE PARTNO=? AND LEVEL &lt;
?&lt;/</assign>
        <assign to="." from="*"></assign>
      </output>
      <input message="inmsg">
        <assign to="." from="*"></assign>
      </input>
    </operation>

  </sequence>
</process>
```

XML Input File for Query

The following example shows how the XML input file from company B might look:

```
<?xml version="1.0" encoding="UTF-8"?>
<A_Query>
  <part_number>12345</part_number>
   <level>10</level>
</A_Query>
```

WSDL for Inventory_Check Web Services Group

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="Inventory_Check"
```

```
    targetNamespace="http://www.sterlingcommerce.com/mesa"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:mesa="http://www.sterlingcommerce.com/mesa"
    xmlns:mesa_xsd="http://www.sterlingcommerce.com/mesa/schema"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <wsdl:types>
        <xs:schema attributeFormDefault="unqualified"
            elementFormDefault="qualified"
            targetNamespace="http://www.sterlingcommerce.com/mesa/schema"
            xmlns="http://www.sterlingcommerce.com/mesa/schema"
            xmlns:tns="http://www.sterlingcommerce.com/mesa/schema"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
            <xs:complexType name="Binary">
                <xs:simpleContent>
                    <xs:extension base="xs:base64Binary">
                        <xs:attribute name="href" type="xs:anyURI"/>
                    </xs:extension>
                </xs:simpleContent>
            </xs:complexType>
            <xs:element name="attachment" type="tns:Binary"/>
            <xs:element name="inlineAttachment" type="xs:base64Binary"/>
            <xs:complexType name="ProcessData">
                <xs:sequence>
                    <xs:any/>
                </xs:sequence>
            </xs:complexType>
            <xs:element name="mesaFault" type="tns:MESAFault"/>
            <xs:complexType name="MESAFault">
                <xs:sequence>
                    <xs:element name="code"/>
                    <xs:element name="message"/>
                    <xs:element name="statusReport"/>
                </xs:sequence>
            </xs:complexType>
            <xs:element name="processData" type="tns:ProcessData"/>
            <xs:element name="documents">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element maxOccurs="unbounded" ref="tns:attachment"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:simpleType name="HashType">
                <xs:restriction base="xs:string">
                    <xs:enumeration value="MD5"/>
                    <xs:enumeration value="NONE"/>
                </xs:restriction>
            </xs:simpleType>
            <xs:complexType name="MESAAuth">
                <xs:sequence>
                    <xs:element name="principal"/>
                    <xs:element name="auth">
                        <xs:complexType>
```

```
                        <xs:simpleContent>
                            <xs:extension base="xs:string">
                              <xs:attribute name="hashType" type="tns:HashType"/>
                            </xs:extension>
                        </xs:simpleContent>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
        <xs:element name="mesaAuth" type="tns:MESAAuth"/>
        <xs:element name="Bead_inquiry" type="tns:ProcessData"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
    </xs:schema>
</wsdl:types>
<wsdl:message name="MESAResponse">
    <wsdl:part element="mesa_xsd:processData" name="parameters"/>
    <wsdl:part element="mesa_xsd:attachment" name="attachment"/>
</wsdl:message>
<wsdl:message name="Bead_inquiry">
    <wsdl:part element="mesa_xsd:mesaAuth" name="header"/>
    <wsdl:part element="mesa_xsd:Bead_inquiry" name="parameters"/>
    <wsdl:part element="mesa_xsd:attachment" name="attachment"/>
</wsdl:message>
<wsdl:message name="MESAFault">
    <wsdl:part element="mesa_xsd:mesaFault" name="parameters"/>
</wsdl:message>
<wsdl:message name="GISGeneric">
    <wsdl:part element="mesa_xsd:mesaAuth" name="header"/>
    <wsdl:part element="mesa_xsd:processData" name="parameters"/>
    <wsdl:part element="mesa_xsd:attachment" name="attachment"/>
</wsdl:message>
<wsdl:portType name="GISPortType">
    <wsdl:operation name="executeBead_inquiry">
        <wsdl:input message="mesa:Bead_inquiry"/>
        <wsdl:output message="mesa:MESAResponse"/>
    </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="GISBinding" type="mesa:GISPortType">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="executeBead_inquiry">
        <soap:operation soapAction="sii:Bead_inquiry"/>
        <wsdl:input>
            <mime:multipartRelated>
                <mime:part>
                    <soap:body parts="header parameters" use="literal"/>
                </mime:part>
                <mime:part>
                    <mime:content part="attachment"
type="application/octetstream"/>
                </mime:part>
            </mime:multipartRelated>
        </wsdl:input>
        <wsdl:output>
            <mime:multipartRelated>
                <mime:part>
                    <soap:body parts="parameters" use="literal"/>
```

```
                    </mime:part>
                    <mime:part>
                        <mime:content part="attachment"
type="application/octetstream"/>
                    </mime:part>
                </mime:multipartRelated>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="Inventory_Check">
        <wsdl:port binding="mesa:GISBinding" name="GISPort">
            <soap:address location="null?service=Inventory_Check"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

Index

U

W

X