Sterling Integrator®

Web Services

Version 5.1

Sterling Commerce
An IBM Company

Contents

Web Services - Overview

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. 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), used to describe Web services in Sterling Integrator. In addition, XML schemas define the structure and content of the XML documents that are shared.

The WSDL contains the information necessary to invoke a Web service, allowing others to access it. It includes (but is not limited to):

- The endpoint URL of the Web service
- The protocols accepted
- Expected input and the data types for each operation
- Output data type for each operation

Sterling Integrator provides the following Web service functionality:

- Ability to be both Web service consumer and provider
- Seamless interoperability with common Web services consumers, supporting .Net 1.1 or 2.0, Axis 1.*x* or 2.0, XFire 1.2.6, or Java EES.
- Ability to create a new Web service
- Ability to generate and view WSDL
- Ability to check in and validate third-party WSDL
- Ability to check in and validate WSDL with HTTPS binding, and use it to create new services (Dynamic Service Creation, now platform-independent)
- HTTPS supported as transport binding
- Ability to create input and output schemas and associate them with a business process
- Message reliability – Messages are received only once
- Compliance with WS-I Basic Profile 1.1 and Basic Security Profile 1.0
- Message-level and transport-level security

- XML encryption and decryption supported
- Encryption algorithm encrypts complete contents of SOAP message body
- Public key encryption supported
- Security for sending and receiving documents, including authentication of users
- Flexible ordering of signature and encryption

- Validation of SOAP message against input and output schemas
- WSDL validation for WSDL checked into Sterling Integrator

Web Services Consumers and Providers

Web services for Sterling Integrator enables you to be both a Web service consumer and a Web service provider.

The following examples describe the Web services consumer and Web services provider:

- Web services consumer – As a consumer, Sterling Integrator consumes external Web services and provides a way to check in external WSDL and generate dynamic services that can be used in a business process to generate a SOAP request.

  Example: While browsing the UDDI directories, you locate a Web service you want to use. You can download the WSDL file that describes it. Next, you would need to check in and provide a version to the WSDL file in Sterling Integrator. When the business process uses 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 – The application, as a Web service provider, exposes Sterling Integrator business processes, services, or both as Web service endpoints.

  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 effectively plan their production for the upcoming days or weeks. 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.

Security Implementation and Web Services

Sterling Integrator builds upon existing security technologies, such as XML digital signatures and X.509 certificates, to deliver an industry-standard way of securing Web services message exchanges. This assures message integrity and prevents security breaches such as replay attacks.

Web services includes the following security features:

- Transport-level security – A secure SSL connection between sender and receiver
- Message-level security – End-to-end security of the SOAP message, regardless of transport-level security or transmission through intermediaries, based on compliance with the WS-Security 1.0 specification (including WS-I Basic Profile 1.1 and Basic Security Profile 1.0).
  - Encryption and decryption features: Encryption of the complete contents of the SOAP message body part, public key encryption, and flexible ordering of signature and encryption.

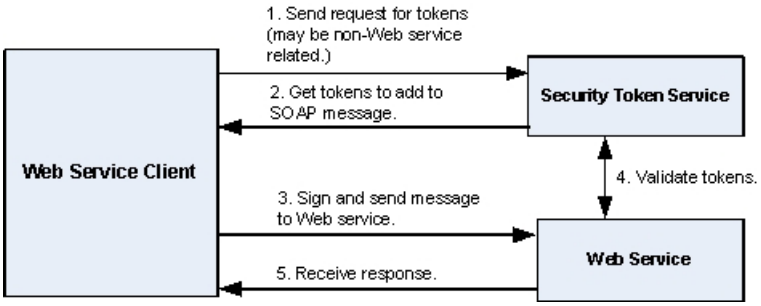
  Note: It does not support shared secret key encryption.

  - Signature creation and verification to authenticate the message sender
  - Processing of security tokens (Username Token and X.509 Certificate Tokens)
  - Security timestamps

For more information about the parameters, refer to the topic *Create a New Web Service* on page 31.

Security Token Flow

The following diagram illustrates the flow of a security token in Sterling Integrator:



UserName Token

You can create a plain text (UserName) security token to and identify yourself when sending and receiving documents using Web services. If you are using a UserName token in your Web service, you need to add it in Sterling Integrator before you configure the Web service. Additionally, you have the option of creating a digested security token. A digested security 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 Sterling Integrator. However, security tokens imported or created using the Web services security token feature are not listed on the Sterling Integrator certificates page.

X.509 Certificate Token

When you configure a Web service in your 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, you can add a security header and a certificate token, which 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.

Implementation Decisions

When making decisions about the security of your Web services, you should consider the following:

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

Message Reliability in Web Services

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

The reliability feature of provides the following features:

- Delivers messages once and in the correct order.
- Processes Reliable Messages (RM) with large attachments.
- Supports SOAP 1.1 and SOAP 1.2 message types.
- Supports synchronous Request-Response Message Exchange Pattern. Sterling Integrator sends back a SOAP response against a SOAP request.
- Responds only for Reliable Message (RM)-Reply Pattern. For unsupported message types, such as Callback, Poll, and so on, the RM-Reply Pattern will send a Feature Not Supported error message back to the user.
- Includes RM Response element as a SOAP header in outgoing SOAP responses.
- Supports Quality of Service (QoS) standard. Messages with any other combinations will result in an exception message back to the originator. The following table lists valid QoS 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

- • Includes message configuration options:
  - Reliable – Only reliable messages are accepted
  - Non-reliable – Only non-reliable messages are accepted
  - AutoDetect – Both types of messages are accepted

Business Processes - Queued and Non-Queued Processing Models

Sterling Integrator supports two modes for business processes: BP Queued processing and BP Non-Queued processing. It is important to understand the difference between the two modes:

- BP Queued mode (default mode) – The process is placed on a business process queue and an available thread executes it. This thread is not the same as the caller thread. BP Queued mode takes advantage of the Sterling Integrator load balancing and scalability features.
- BP Non-Queued mode – All of the steps in the business process run on the same thread as the caller thread (or the thread where the bootstrapping of the business process has taken effect). This thread will only be released when the business process completes. If there are multiple business processes that need to be bootstrapped by a service or adapter, the processes are executed one at a time until they have all completed. Consequently, if you choose to run in BP Non-Queued mode, load balancing is not supported and there may be potential for performance issues.

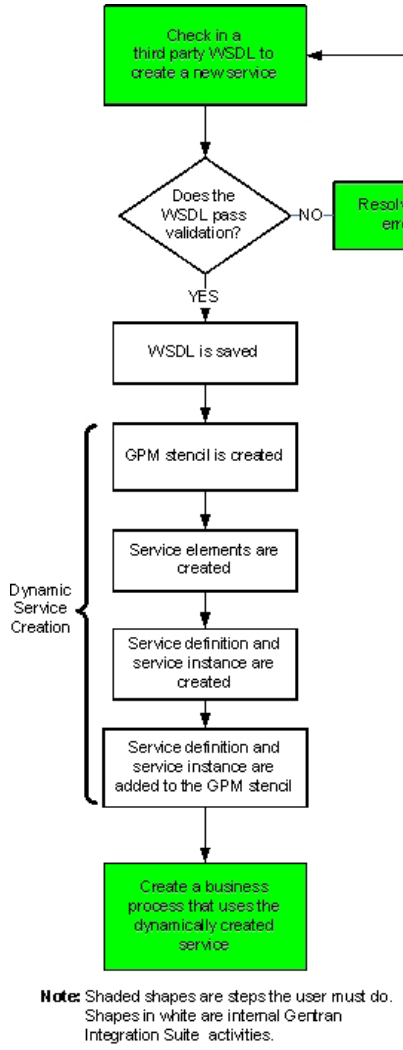
Predominantly, business processes need to be configured in BP Queued mode to utilize the system capabilities. The following are cases where business processes can be processed in BP Non-Queued mode:

- If the business process needs to be processed immediately before the calling adapter finishes its processing executing as part of the caller thread
- If the business process level transactions need to be part of the parent transaction

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 application as a Web services consumer supports either a default configuration with HTTP transport, or a manual configuration with either HTTP or HTTPS transport.

The following figure illustrates the Dynamic Service Creation process:



Transport Options

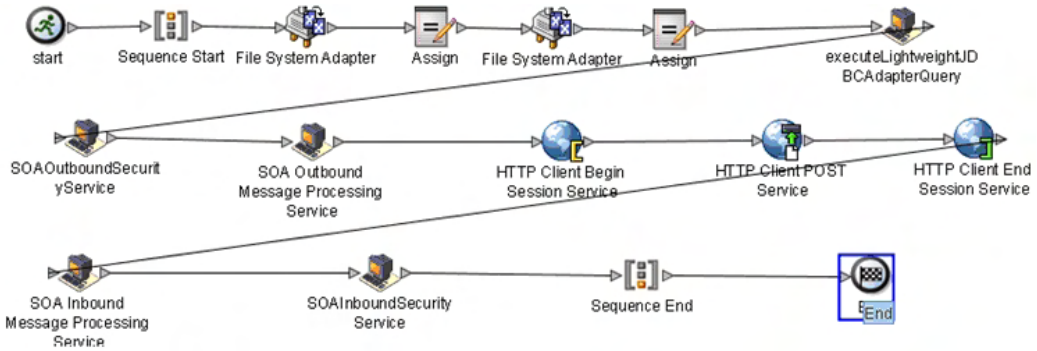
When the WSDL is checked in, you can select either of the two transport options for the Dynamic service that is created:

- A legacy configuration that calls services and adapters at the API level, and is restricted to use of HTTP for transport.
- A transport-independent implementation that restricts the role of dynamic services to generate the SOAP message from the checked-in WSDL file allowing use of either HTTP or HTTPS adapter.

  This option requires manual configuration in the Graphical Process Modeler (GPM) of the SOA Inbound and SOA Outbound message processing services, (optionally) the SOA Inbound and SOA Outbound Security services, and the adapters used for transport.

As a comparison, the GPM configuration of the two transport options could appear as follows. This is an example with the default HTTP transport configuration:

The following is an example of a GPM configuration for a transport-independent (HTTP/HTTPS) configuration:



For more information about the Dynamic Service Creation process, refer to the *Dynamic Services* documentation.

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, which can be either simple or 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

Input and Output Schemas

Every business process that can be run by a Web services provider can be associated with an output and/or an input schema:

- The input schema is a schema object (XSD) that defines the structure of the XML elements present in the body of the incoming SOAP request. This element in the SOAP body is inserted "as is" into the process data of the business process to be run.

  The input schema is also used in the WSDL generated for the Web service hosting the business process, as a type definition for the input part of the operation corresponding to the business process.
- The output schema is a schema object (XSD) that defines the structure of the XML elements to be sent as the body of the outgoing SOAP response. This element is extracted from the process data of the business process that was run by the Web service provider and inserted into the SOAP body of the response. Mapping to an output schema provides a way, if desired, to restrict what process data is passed to the consumer in the SOAP response.

  The output schema is also used as a type definition in the WSDL generated for the Web service hosting the business process, in the output part of the operation corresponding to the business process.

Sterling Integrator provides default input and output schemas. The default schemas do not contain any restrictions on the structure or type of data in the message, but are included for use by the application if no other schema is provided.

- Default input schema – Contains two parameters, process data and primary document.
- Default output schema – Contains one parameter, process data. It passes the entire process data in the response.

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.

The user can also select whether to validate incoming or outgoing data, each against its respective schema.

Naming Conventions for Schemas

When creating schemas, you should provide descriptive names that include information such as the business function or consumer name/type. Additionally, include the direction (input or output) if the schema will be used for only one direction.

Business Process Schema Limitations

Business process schemas have the following limitations:

- Each business process can have only one root element mapped to it per input or output schema.
- Valid schemas can have one or more root elements.
- 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.
- Target namespace must be present in the schema for the WSDL to generate properly.

Note: You must add an XML schema to the application before you can map it to an existing business process.

Web Services Schema Limitations and Structure

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 Web services structure must include the following:

Node	Description
Type of data	(Required) Accepts String and other types. Default is String.
Document encoding type	(Required) Accepts UTF8 and UTF16 (SOAP specification). Default is UTF8.

WSDL Example

If you check in an output schema and select the use of the output schema during Web service creation, the elements of the schema are inserted into the “types” section of the WSDL. The following example shows the types section of a WSDL with schema elements inserted:

```
...
<wsdl:definitions ...
<wsdl:types> ...
<xs:schema ...
…
<xs:complexType name="CustomElement">
 <xs:sequence>
  <xs:element name="ele1" type="xsd:string"/>
  <xs:element name="ele2" type="xsd:string"/>
  <xs:element name="ele3" type="xsd:string"/>
 </xs:sequence>
```

```
</xs:complexType>
<xs:element name="outputData" type="tns: CustomElement" />
...
```

The type is then inserted into the definition of the output message of the corresponding operation. This is done in a new message that corresponds to the business process. The syntax for the unique name of the message element consists of the name of the business process concatenated with the word “Response”:

```
business_process_nameResponse
```

The following example shows the syntax:

```
...
<wsdl:message name="BP1Response">
 <wsdl:part element="mesa: CustomElement" name="outputData" />
</wsdl:message>
...
<wsdl:portType name="GISPortType">
 <wsdl:operation name="executeBP1">
  <wsdl:input message="mesa:BP1" />
  <wsdl:output message="mesa: BP1Response" />
 </wsdl:operation>
</wsdl:portType>
```

WSDL Validation

During the WSDL check in process, Sterling Integrator automatically validates the WSDL file.

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.

The WSDL file is validated against the following rules:

- 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 a non-XML file.
- WSDL Syntax – Sterling Integrator 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 valid 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. Sterling Integrator dynamic services supports only SOAP binding over the HTTP or HTTPS 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 known issue in Sterling Integrator and the WSDL is internally parsed.
  - Document/Literal WSDL duplicate operation names are never allowed.

Web Service Configuration via URL

During Web service configuration, a base URL is provided for the Web service. The URL is present in the WSDL file and can be viewed through the Generate WSDL feature. 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 can locate and configure many components that are already part of Sterling Integrator. You can use almost any business process, service, or adapter (except internal services and business processes) as a Web service component.

Consider the following as you plan a new Web service:

Item	What to Consider
Services or adapters	Which services or adapters could this new Web service invoke when a consumer request is received? Hint: Generally, it is recommended to create a unique configuration of any service or adapter for use with your Web service.
Business processes	Which services or adapters could this new Web service invoke when a consumer request 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. In addition, all services and adapters are enabled.
Transport	Do I want to use standard HTTP or secure HTTPS for the SOAP transport binding for this Web service?
Security settings	Do I want to use legacy security settings for backward compatibility with existing business processes or enhanced security settings that offers enhanced message-level security, including encryption/decryption, signing options, and timestamps? **Note:** Legacy security settings, business processes, and services will be phased out in subsequent releases.
Signing / Encryption Order	Determine the order of signature and encryption (Sign First or Encrypt First).

Item	What to Consider
Queued Mode	Do I want to queue business processes or execute them on the same thread as the thread where the bootstrapping of the business process has taken effect?
Input XML Schema	Do I want Sterling Integrator to use a specific input schema when processing the SOAP request? Hint: Associate an input XML schema with a business process and then associate the business process with the Web service.
Output XML Schema	Do I want a customized or default response from Sterling Integrator? Hint: Associate an output XML schema with a business process and then associate the business process with the Web service.
Message Reliability Settings	What type of messages will my Web service accept? Sterling Integrator supports the following reliability settings: • AutoDetect • ReliableOnly • NonReliable **Note:** You cannot send the same reliable message (groupID and sequenceNo) to two different Web services configured in the same Sterling Integrator instance.
WS-I Conformance Settings	What type of message conformance will my Web service have? Sterling Integrator supports the following conformance types: • Conform WSDL – Adds a WS-I claim element to every WSDL node • Conform SOAP Response – Adds a WS-I claim header to the SOAP response
Attachment Settings	What type of attachments will my Web service accept? Sterling Integrator supports the following types of attachments: **Note:** Attachments do not need to be XML. • Input – Request can have an attachment as part of SOAP message. • Output – Sends the primary document as an attachment along with the SOAP response as a multipart MIME message. Optional. **Restriction:** Microsoft® products support only dime format, not mime or s-mime. • Inline – No attachments. Information must be contained inline in the message.

Item	What to Consider
	Example: A multipart mime message. Optional. **Note:** If the Inline option is selected, the resulting WSDL will not be compliant with WS-I Basic Profile 1.0. See the topic FAQs About Web Services Interoperability for additional information on attachments.
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 Sterling Integrator response to a consumer • Encryption Certificate – Used to encrypt the body of the response
Consumer Access to the Web Service	In order for a consumer to access and use your Web service, you must create a user account for them in Sterling Integrator. The user account must have access rights to all components used with the Web service (services, adapters, and business processes).

Checklist: Develop a New Web Service

Once you have identified a need to provide a new Web service, the following checklist guides you through the implementation process:

1. Plan the business function.
2. Identify the consumer who requires access to this Web service (your partner). Create a user account for each consumer in Sterling Integrator and send the user account and password to the consumer.
3. Fill out the Collecting Consumer Information and Collecting Provider Information checklists. The checklists keep information relevant to developing and implementing a new Web service in two documents.
4. Discuss the following security requirements with your Web service consumer:
   - Will messages be signed?
   - Will messages be encrypted?
   - What type of certificate will they use (CA or other third party)?
   - What type of encryption will be used, and in what order (encryption or signature first)?
   - When will certificates be obtained?
   - How will the public keys and passphrases be exchanged (e-mail, phone, or letter)?
5. Create a system certificate, if responses must be signed, and send the consumer the public key for the certificate.
6. Check the certificate into Sterling Integrator.
7. If requests must be signed, have the consumer obtain a CA certificate and send you the public key.
8. Check the public key for your consumer's certificate into Sterling Integrator.
9. Decide which components (services, adapters, or business processes) you need to create in Sterling Integrator for this Web service.
10. Create the necessary service and adapter configurations and business process models. Ensure that the service configurations are enabled.

    Note: If you plan to use HTTPS instead of HTTP, select "https" in the SOAP Transport Bindings Settings page during configuration of the SOA HTTP Server adapter instance.

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 version.

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 are used. The default schemas do not perform any typing or validation.

13. Check the XML schemas into Sterling Integrator.
14. Create a Web service in Sterling Integrator.
15. Map the XML schemas to the appropriate business process models in Sterling Integrator.
16. Generate the WSDL for the Web service.
17. Send the WSDL to your consumers or enable them to access it from a URL on the Sterling Integrator Web server. The consumer needs the WSDL to determine how to create SOAP messages.

    Note: To publish the Web service to a UDDI, use a third party tool.

18. Test the Web service.
19. Once testing is complete, notify the consumer that the service is ready for use.

Collect Consumer Information for a New Web Service

Collect the following consumer 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 messages to be 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): ____/____/____

Item	Description
	Certificate Name/ID: _____
Testing	WSDL URL tested on (date): ____/____/____ Comments: _____ _____ _____ Web service URL tested on (date): ____/____/____ Comments: _____ _____ _____

Collect Provider Information for a New Web Service

Collect the following provider 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 messages to be 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): ____/____/____

Item	Description
	Certificate Name/ID: _____
Order of signing / encryption	Sign first or encrypt first?_____
Testing	WSDL URL tested on (date): ____/____/____ Comments: _____ _____ _____ Web service URL tested on (date): ____/____/____ Comments: _____ _____ _____

Web Service Settings for Enhanced Security and Interoperability

This topic covers the following Web service settings for enhanced security and interoperability:

- Enhanced Settings
- Legacy Settings

Enhanced Settings

When creating a new Web service, choose the enhanced security and/or interoperability settings below to conform to the following specifications. You may choose to conform to either or both specifications. All parameters are independent.

- WS-I Basic Profile (WS-I BP) – Conforming WSDL and SOAP responses for enhanced client interoperability.
- WS-I Basic Security Profile (WS-I BSP) – Provides exchange of certificates, encryption, timestamps, and other enhanced security features.

For backward compatibility, refer to *Legacy Settings* on page 29.

A complete description of all options is included in the task *Create a New Web Service* on page 31.

Page	Parameter	Applicable Specification	Conforming Option or Value
Web Service: Name	Use legacy WS settings (for backward compatibility)	WS-I BSP 1.0	Clear the check box.
Response Security Settings	**Signature Settings** – If a Signing Certificate is selected:		
	Signing Key Identifier	WS-I BSP 1.0	SubjectKeyIdentifier is the ONLY option that conforms to WS-I BP 1.0.
	CanonicalizationAlgorithm	WS-I BSP 1.0	C14n_Excl_Omit_Comments and C14n_Excl_With_Comments conform to WS-I BP1.0.

Page	Parameter	Applicable Specification	Conforming Option or Value
			Other options may fail if the canonicalization is not exclusive.
	Encryption Settings – If EncryptionCertificate is selected:		
	EncryptionKeyIdentifier	WS-I BSP 1.0	SubjectKeyIdentifier is the only option compliant with WS-I BSP1.0
	Symmetric Algorithm	WS-I BSP 1.0	AES-192 CBC does NOT conform. All other options conform.
Assign Consumers	(Selection box)		Do NOT select if you want WSDL to conform to WS-I BP 1.1. **Note:** Selection of Consumers does NOT affect conformance of SOAP response.
WS-I Basic profile 1.1 Conformance Settings	Make the generated WSDL conform to WS-I Basic Profile 1.1	WS-I Basic Profile 1.1	Select the check box.
	Make the SOAP Response conform to WS-I Basic Profile 1.1		Select the check box.
Attachment Settings **Restriction:** You cannot send or receive attachments in Sterling Integrator with .NET.	Input attachment	WS-I Basic Profile 1.1	Either Output or Input attachment allows WSDL to conform to WS-I BP 1.1.
	Output attachment		
	Inline attachment		Do NOT select if you want to make the WSDL conform to WS-I BP 1.1.

Legacy Settings

The default legacy settings are available for backward compatibility with existing business processes and services. While a legacy setting does not allow WS-I Basic Security Profile conformance, it does allow you to maximize interoperability by generating WSDL and SOAP responses that conform to WS-I BP1.1.

Note: Legacy settings will eventually be retired in favor of the Enhanced settings.

Page	Parameter	Applicable Specification	Conforming Option or Value
Web Service: Name	Use legacy WS settings (for backward compatibility)		Accept default
Assign Consumers	(Selection box)	WS-I BP 1.1	Do NOT select if you want WSDL to conform to WS-I BP 1.1. **Note:** Selection of Consumers does NOT affect conformance of SOAP response.
WS-I Basic profile 1.1 Conformance Settings	Make the generated WSDL conform to WS-I Basic Profile 1.1	WS-I BP 1.1	Select the check box.
	Make the SOAP Response conform to WS-I Basic Profile 1.1		Select the check box.
Attachment Settings **Restriction:** You cannot send or receive attachments in Sterling Integrator with .NET.	Input attachment	WS-I BP 1.1	Selection of either Output or Input attachment allows WSDL to conform to WS-I BP 1.1.
	Output attachment		
	Inline attachment		Do NOT select if you want to make the WSDL conform to WS-I BP 1.1

Create a New Web Service

To create a new Web service:

- From the **Deployment** menu, select **Web Services** > **Manager**.
- In the Web Services Management page, under **Create**, next to **Create a Web Service Configuration**, click **Go!**
- In the Web Service : Name page that is displayed, enter information in the following fields and click **Next**.

Field	Description
Name	(Required) Unique name for the Web service.
Description	(Required) Description for the Web service.
Use legacy WS settings (for backward compatibility)	Select the check box to use legacy security settings and existing business processes and services, and to ensure compatibility with existing instances. Legacy settings allow the selection of WS-I Basic Profile 1.1-compliant SOAP responses and generated WSDL (default). Clear the check box to use enhanced WS-I Basic Security Profile 1.0 security features and to maximize interoperability with most Web services consumer types with WS-I Basic Profile 1.1 settings (recommended). For more information, refer to the topic *Web Service Settings for Enhanced Security and Interoperability* on page 28 **Note:** Legacy business processes and services will eventually be retired.
Do not use BP queued mode	In BP queued mode (default mode), the process is placed on a business process queue and executed by an available thread. BP Queued mode is preferred. Select the **Do not use BP Queued mode** check box to use BP Non-Queued mode. In BP Non-Queued mode, the steps in the business process run on the same thread as the thread where the bootstrapping of the business process has taken effect. The thread is only released when the

Field	Description
	business process completes, raising potential performance issues. However, BP Non-Queued mode may be preferred in these situations: • If the business process needs to be processed immediately before the calling adapter finishes its processing as part of the caller thread • If the business process level transactions need to be part of the parent transaction

- In the SOAP Transport Binding Settings page that is displayed, enter information in the following fields and click **Next**.

Field	Description
Use HTTP as SOAP Transport	Uses the standard binding, HTTP, as the transport protocol.
Use HTTPS as SOAP Transport	Uses HTTP with SSL as the transport protocol for secure transport.

- In the Request Security Settings page that is displayed, enter information in the following fields and click **Next**.

  Fields are displayed based on the security settings selected on the Web Service Name page: legacy or enhanced.

  The following fields are applicable when the **Use legacy WS settings (for backward compatibility)** check box is selected.

  Legacy Settings

Field	Description
Verification Certificate	To use a verification (Trusted) certificate with the security header, click the list icon. Select a certificate and click **Save**. Required if Username token is not used and Security Header is Yes.
Security Header	(Optional) Select **Yes** if you want to include a security header in the request. If selected, a verification certificate or UserName token, or both, must be added. Default is No.
UserName Token	Select the check box to add a UserName (Security) token with the security header. Required if Security Header is yes and verification certificate is not used.
UserName Tokens	Select a UserName (Security) token from the drop-down list. Required if UserName Token is selected.

  Enhanced Security Settings

The following fields are applicable when the **Use legacy WS settings (for backward compatibility)** check box is not selected, that is, when you are using enhanced security settings:

Field	Description
Verification Certificate	To use a verification (Trusted) certificate with the security header, click the list icon. Select a certificate and click **Save**. Required if Username token is not used.
Decryption Certificate	Select a decryption certificate to decrypt the encrypted body in an incoming request. Click the list icon to access available system certificates. Select a certificate from the list and click **Save**.
UserName Tokens	Select a UserName (Security) token from the drop-down list.

- In the Response Security Settings page that is displayed, enter information in the following fields and click **Next**.

  Fields are displayed based on the security settings selected on the Web Service Name page: legacy or enhanced.

  The following fields are applicable when **Use legacy WS settings (for backward compatibility)** check box is selected.

  Legacy Settings

Field	Description
Signing Certificate	To use a signing (System) certificate with the security header, click the list icon. Select a certificate and click **Save**. Required if Security Header is Yes. **Note:** RSA certificates can be created through the user interface (**Trading Partner** > **Digital Certificates** > **System**). To generate a DSA certificate, use the `CreatCertEx` script installed along with Sterling Integrator.
Security Header	(Optional) Select **Yes** if you expect a security header in the response. If selected, a signing certificate (X.509) or UserName token, or both, must be added.
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.

Field	Description
	• X509KeyIdentifier – Sends the X.509 certificate used to encrypt the symmetric key.
UserName Token	To use a Username token with the security header, select the check box and select a UserName token from the drop-down list. Required if Security Header is Yes.
UserName Tokens	Select a UserName (Security) token from the drop-down list. Required if UserName Token is selected.

Enhanced Security Settings

The following fields are applicable when **Use legacy WS settings (for backward compatibility)** check box is not selected, that is, when you are using enhanced security settings:

Field	Description
WS-Security Header Settings	
Mustunderstand (INSERT_MUSTUNDERSTAND)	Select the check box to add this attribute to the Security header and make processing mandatory for the receiving party.
Actor: (SECURITY_HEADER_ACTOR)	Recipient or other appropriate Actor name against which the Security header element will be created in the outgoing SOAP Request.
UserNameToken (INSERT_USER_NAME_TOKEN)	Select the check box if you want a user name token in the Security Header.
UserName Tokens: (USE_SECURITY_TOKEN)	Select an available username token (security token) to insert in the outgoing SOAP request from the drop-down list.
Insert Time Stamp (INSERT_TIME_STAMPS)	Select the check box to allow insertion of creation and expiration times in a message in accordance with WS-Security specification 1.0. This allows the recipient of a SOAP message to decline processing a WS-Security header if its timestamps are expired.
TimeStamp Interval (TIME_TO_LIVE)	Maximum length of time in seconds for which the timestamp is considered valid. Valid value is any integer. Default is 0.
SignatureSettings	
Signing Certificate (SIGN_WITH_KEY)	To use a signing (System) certificate with the security header, click the list icon to access available certificates. Select a certificate from the list and click Save.

Field	Description
	Note: RSA certificates can be created through the user interface (**Trading Partner** > **Digital Certificates** > **System**). To generate a DSA certificate, use the `CreatCertEx` script installed along with Sterling Integrator.
Signing Algorithm (SIGNING_ALGO)	Select the name of the algorithm to use for signing the SOAP message. Required if a Signing Certificate is selected. Options and locations for more information are: • RSA – *http://www.w3.org/2000/09/xmldsig#rsa-sha1* • DSA – *http://www.w3.org/2000/09/xmldsig#dsa-sha1*
SigningKeyIdentifier (X509_CERTIFICATE_OPTION)	Select a type to designate how key information is carried with an encrypted document to allow decryption. Required if a Signing Certificate is selected. Valid values: blank, BinaryToken, IssuerSerial, X509KeyIdentifier, or SubjectKeyIdentifier **Note:** To be compliant with WS-I BSP 1.0, select SubjectKeyIdentifier
CanonicalizationAlgorithm (CANONICALIZATION_ALGO)	Select algorithm to use to canonicalize the document before signing the body of the response. Required if a Signing Certificate is selected. These are options and locations for more information. • C14n_Omit_Comments – *http://www.w3.org/2001/10/xml-exc-c14n* • C14n_With_Comments – *http://www.w3.org/2001/10/xml-exc-c14n#WithComments* • C14n_Excl_Omit_Comments – *http://www.w3.org/TR/2001/REC-xml-c14n-20010315* • C14n_Excl_With_Comments – *http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments* **Note:** To be compliant with WS-I BSP 1.0, select C14n_Excl_Omit_Comments or C14n_Excl_With_Comments.
EncryptionSettings	
EncryptionCertificate	Select the certificate to use to encrypt the body of the response and click **Save**.
KeyEncodingAlgorithm (KEY_ENCODING_ALGO)	Select a key encoding algorithm to be used for encryption based on the public key of the recipient. Required if an Encryption Certificate is selected. The options and location for more information are: • RSA15 – *http://www.w3.org/2001/04/xmlenc#rsa-1_5* • RSAOEP – *http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p*

Field	Description
	Note: Both options require an RSA public key certificate.
EncryptionKeyIdentifier (ENCRYPT_KEY_IDENTIFIER_TYPE)	Select a key identifier type to use for encryption. Required if an Encryption Certificate is selected. **Important:** SubjectKeyIdentifier is the only option that is compliant with WS-I BSP1.0. • BinaryToken – Sends the encrypting 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. • SubjectKeyIdentifier – Only option that is compliant with WS-I BSP1.0. An X.509 v3 certificate extension that should be present in the encryption certificate used. If this extension is not present in the certificate, this option should not be selected.
SymmetricAlgorithm (SYMMETRIC_KEY_ALGO)	Select a symmetric key algorithm to use for encryption.Required if an Encryption Certificate is selected. These are the options and locations for more information. **Important:** AES-192 CBC is **NOT** compliant with WS-I BSP1.0. • Triple-DES CBC: *http://www.w3.org/2001/04/xmlenc#tripledes-cbc* • AES-128 CBC: *http://www.w3.org/2001/04/xmlenc#aes128-cbc* • AES-192 CBC: *http://www.w3.org/2001/04/xmlenc#aes192-cbc* – Not compliant with WS-I BSP1.0. • AES-256 CBC: *http://www.w3.org/2001/04/xmlenc#aes256-cbc* **Note:** Use of AES-192 or AES-256 requires updating the JCE Policy files that come with the JRE with unlimited strength cryptography. For more information about JCE policy files, refer to the Sterling Integrator Installation documentation.
SigningEncryptionOrder (SIGNATURE_ENCRYPTION_ORDER)	Select the order from the drop-down list in which signature and encryption will be done, based on the decision reached with your other party. Valid values are Sign First and Encrypt First.

- In the Assign Business Processes page that is displayed, enter information in the following fields and click **Next**.

Field	Description
Filter by name	(Optional) To filter the list of available business processes, type part of the business process name into the Filter field and click the **Filter** icon.
Available and Selected lists	(Optional) Select one or more business processes from the list of available business processes on the left to be associated with this Web service and click the right arrow. To select all available business processes, click the double right arrow. A business process used as part of a Web service must be the default version of that business process. **Note:** Assigning a business process (as opposed to assigning a service and populating it outside of a business process) is a best practice.

- In the Assign Services Instances page that is displayed, enter information in the following fields and click **Next**.

Field	Description
Filter by name	(Optional) To filter the list of available services, type part of the service name into the Filter field and click the **Filter** icon.
Available and Selected lists	(Optional) Select one or more services from the list of available services on the left to be associated with this Web service and click the right arrow. To select all available services, click the double right arrow. **Note:** Assigning a business process (as opposed to assigning a service and populating it outside of a business process) is a best practice.

- In the Assign Consumers page that is displayed, enter information in the following fields and click **Next**.

Field	Description
Filter by name	(Optional) To filter the list of available user accounts, type part of the user name into the Filter field and click the **Filter** icon.
Available and Selected lists	(Optional) Select one or more users from the list of available users on the left to be associated with this Web service and click the right arrow. To select all available services, click the double right arrow. **Note:** If you select a consumer, the mesaAuth element is inserted into the “types” section of the input message of the generated WSDL. The mesaAuth element is a

Field	Description
	mechanism internal to Sterling Integrator, and the resulting WSDL is not compliant with WS-I Basic Profile 1.1. If you do not select a consumer, the mesaAuth element is not inserted and the output is WS-I compliant.

- In the Reliability Settings page that is displayed, enter information in the following fields and click **Next**.

Note: You cannot send the same reliable message (with the same groupID and sequenceNo) to two different Web services configured in the same instance of Sterling Integrator, or the operation will fail.

Field	Description
Reliability Settings	(Required) Select one of the following options. • AutoDetect – The Web service end point detects the RM header in the incoming message (default). **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 and processed by the WS end point. • NonReliableOnly – Only non-reliable messages are accepted.

- In the WS-I Basic profile 1.1 Conformance Settings page that is displayed, enter information in the following fields and click **Next**.

Field	Description
Make the generated WSDL conform to WS-I Basic Profile 1.1	(Optional) Adds a WS-I claim element to every WSDL node. **Note:** Not available if any Consumers have been selected. If the setting is disabled and you want to select it, go back and remove all consumers (users) from the Assign Consumers screen.
Make the SOAP Response conform to WS-I Basic Profile 1.1	(Optional) Adds a WS-I claim element to the header of a SOAP response.

- In the Attachment Settings page that is displayed, enter information in the following fields and click **Next**.

Note: You cannot send or receive attachments in Sterling Integrator with .NET.

Field	Description
Input Attachment	(Optional) Request can have an attachment as part of the SOAP message.
Output Attachment	(Optional) Sends the primary document as an attachment with the SOAP response as a multipart MIME message. **Notes:** • Microsoft® products support the dime format, not mime or s-mime. • Do not select if sending a SOAP Outbound response containing multiple attachments. Refer to the *SOAP Outbound service* documentation for information on how to send multiple attachments.
Inline Attachment	(Optional) No attachments. Information must be contained inline in the message. **Caution:** If this option is selected, the resulting WSDL is not compliant with WS-I Basic Profile 1.0.

- In the Confirm page that is displayed, review the changes and perform one of the following tasks:
  - To change the Web service settings, click **Back**.
  - To abandon the Web service created, click **Cancel**.
  - To save the Web service created, click **Finish**.

  To view the base URL for your Web service, refer to the WSDL file, which can be viewed through the Generate WSDL feature.

Test 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 Sterling Integrator 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 Sterling Integrator 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, and so on) 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), or are sent to the correct recipient.

Delete a Web Service

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. In the Web Services Management page, perform one of the following tasks:
   - 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!**
   - Next to **List Alphabetically**, Click **Go!**
3. Select the delete check box 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 is for the correct Web service and click **Delete**.

Generate WSDL for a New Web Service

The WSDL specifies XML data types for elements and attributes, allowing the customer to pass the value for the parameter according to the specified data type.

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

1. From the **Deployment** menu, select **Web Services** > **Manager**.
2. In the Web Services Management 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!**
3. On the **Web services** results page, click **Generate WSDL** next to the name of the Web service.
4. A Confirm page displays. Click **Finish** to generate the WSDL.

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

   `WSDL Generation has been completed successfully`.
5. Perform one of the following tasks:
   - Click **Return**.
   - Click **View WSDL** and then review the WSDL or note the URL. Close the file when completed.
   - Click **Download WSDL** and follow the instructions to download the file to your desktop.
6. Click **Return** to continue.

View 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**.
2. In the Web services manager page, under List, select **All** from the **Alphabetically** drop-down list and click **Go!**
3. Click **View WSDL**.

Make the Web Service Available to Your Users

In order for your users to access the Web service, you must either distribute or publish the generated WSDL for the Web service.

Perform one of the following:

- Distribute the WSDL as a file to your consumers.
- 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 service configuration and WSDL generation)
  • *WebServiceName* is the name of the Web service.

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

Check In a WSDL for a Web Service

To check in a WSDL:

1. From the **Deployment** menu, select **Web Services** > **WSDL Check In**.
2. In the WSDL page, under **Check in**, next to **Check in new WSDL**, click **Go!**
3. In the Naming page that is displayed, enter information in the following fields and click **Next**.

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

4. In the Naming: WSDL Transport Binding Settings page that is displayed, select a protocol to use as the transport binding to send the SOAP message to the Web service provider and click **Next**.

Field	Description
Default Transport (HTTP)	Uses HTTP as the transport binding. This option generates the dynamic service and configures the associated business processes (including message processing and security services) and URIs automatically. Choose for backward compatibility. Uses HTTP services and SOAInbound and SOAOutbound Services internally.
Other Transport (HTTP/HTTPS)	Uses HTTP or HTTP with SSL as the transport binding. This option creates the dynamic service for the checked-in WSDL. Configure the business process as follows: `DynamicService --> SOAOutboundSecurity Service (optional) -->`

Field	Description
	`SOAOutboundMessageProcessingService --> Transport --> SOAInboundMessageProcessingService --> SOAInboundSecurityService (optional)` Uses either an HTTP Server adapter or a new HTTP Server adapter instance (SOA SSL Http Server adapter). You may need to manually configure services. For more information on configuring services, refer to the services documentation. By using this option, you should: • Use the GPM to configure the SOAOutboundMessageProcessing and (if desired) the SOAOutboundSecurityService manually at the business process level for use after Dynamic Service Generation generates the SOAP message. You must provide an end point address and port to use to send the SOAP message to the Web service provider. • Configure the SOAInboundMessageProcessing and (if desired) SOAInboundSecurityService for use after the SOAP message is sent to the Web service provider.

5. In the Naming: Select WSDL page that is displayed, enter information in the following fields and click **Next**.

   Fields are displayed based on the input mode selected for defining the new WSDL on the Naming page.

   The following fields are applicable when **Check in WSDL** check box is selected.

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

   The following fields are applicable when **WSDL Text Editor** check box is selected.

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

6. In the Confirm page that is displayed, select **Enable for Business Processes** check box if you want to use this WSDL for use with business processes. Review the changes and perform one of the following tasks:
    - To change the WSDL Check In settings, click **Back**.
    - To abandon the WSDL Check In settings, click **Cancel**.
    - To check in the new WSDL, click **Finish**.

Check In a New Version of a WSDL

To check in a new version for a WSDL:

1. From the **Deployment** menu, select **Web Services** > **WSDL Check In**.
2. In the WSDL page, under **List**, select **All** from the Alphabetically drop-down list, click **Go!**
3. In the WSDL page, click **source manager** next to the WSDL for which you want to check in a new version.
4. In the WSDL Source Manager page, click **check out** next to the WSDL.
5. When prompted with the following message, click **OK** to check out the file or click **Cancel** to check out a read-only copy of the file:

   ```
   Click OK to lock the file for editing. During the check in process, you can release the lock. Click Cancel to check out a read-only copy of the file.
   ```

6. In the WSDL Source Manager Page, next to **Check in a new version of this WSDL**, click **Go!**
7. In the Naming: WSDL Transport Binding Settings page that is displayed, select a protocol to use as the transport binding to send the SOAP message to the Web service provider and click **Next**.

Field	Description
Default Transport (HTTP)	Uses HTTP as the transport binding. This option generates the dynamic service and configures the associated business processes (including message processing and security services) and URIs automatically. Choose for backward compatibility. Uses HTTP services and SOAInbound and SOAOutbound Services internally.
Other Transport (HTTP/HTTPS)	Uses HTTP or HTTP with SSL as the transport binding. This option creates the dynamic service for the checked-in WSDL. Configure the business process as follows: `DynamicService --> SOAOutboundSecurity Service (optional) --> SOAOutboundMessageProcessingService --> Transport --> SOAInboundMessageProcessingService --> SOAInboundSecurityService (optional)`

Field	Description
	Uses either an HTTP Server adapter or a new HTTP Server adapter instance (SOA SSL Http Server adapter). You may need to manually configure services. For more information on configuring services, refer to the services documentation. By using this option, you should: • Use the GPM to configure the SOAOutboundMessageProcessing and (if desired) the SOAOutboundSecurityService manually at the business process level for use after Dynamic Service Generation generates the SOAP message. You must provide an end point address and port to use to send the SOAP message to the Web service provider. • Configure the SOAInboundMessageProcessing and (if desired) SOAInboundSecurityService for use after the SOAP message is sent to the Web service provider.

8. In the Naming: Select WSDL page that is displayed, enter information in the following fields and click **Next**.

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

9. In the WSDL: Set Default Version page, select the WSDL under the **OTHER Versions** section.
10. In the Confirm page that is displayed, perform the following tasks:
    a) Select **Enable for Business Processes** check box if you want to use this WSDL for use with business processes.
    b) Select **Release the lock on the file** check box to unlock the WSDL file.
11. Click **Finish**.

Delete a WSDL

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

To delete a WSDL file:

1. From the **Deployment** menu, select **Web Services** > **Manager**.
2. Enter the name of the WSDL file in the Search box and click **Go!**

   You can also select the first letter of the WSDL file name from the **Alphabetically** drop-down list and click **Go!**.

   All WSDL files are displayed.
3. Click the WSDL file to be deleted. The WSDL file settings are displayed in a separate pop-up window.
4. In the pop-up window, verify the WSDL file to delete and click **Close**.
5. Select **Delete** next to the WSDL file 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 message is displayed. Verify that the information shown is for the appropriate WSDL file and click **Delete**.

Create an XML Schema for Web Services

To create an input XML schema or output XML schema for Web services:

1. Create the input XML schema or output XML schema for your business process using an XML text editor.
2. Check in the XML schemas into Sterling Integrator.

   Note: You can use only XML schemas (.xsd) with Web services. You cannot use DTD (.dtd) files.

3. Map the XML schemas to the business process in Sterling Integrator.

Map XML Schemas to Business Processes

To map an XML schema to a business process:

1. From the **Deployment** menu, select **Web Services** > **Schema Mappings**.
2. In the BP Schema Configuration page, under **Create**, next to **Create a new BP Schema Mapping**, click **Go!**
3. In the BP Schema Mapping page that is displayed, select the appropriate Business Process, Input Schema, Output Schema from the respective drop-down lists and click **Next**.

Field	Description
Business Process	(Required) If this is a new mapping, select a business process to associate input and/or output schemas with. 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. The generic output XML schema outputs the whole of the process data for the business process as the SOAP response.

4. In the BP Root Element Mapping page that is displayed, the selected Business Process, Input Schema, and Output Schema is displayed. Select the appropriate settings and click **Next**.

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 is previously selected.
Validate with Input Schema	Select the check box if you want to validate the SOAP body content against the input schema. Send as SOAP body content if validation is successful, or send a fault if errors are encountered.
Output Root Element	Select a Root Element for the output schema to define the output formats for a business process. Required if output schema is previously selected.
Validate with Output Schema	Select the check box if you want to validate the content of the response against the output schema. Send as SOAP body content if validation is successful, or send a fault if errors are encountered.

There may be times when sending a blank SOAP body is required. To send a blank SOAP body and have it validate successfully:

- Do not select the **Validate with Output Schema** check box.
- Do not create the WebserviceResponseNode element in process data (or create it and leave the node empty).

5. In the Confirm page, review the settings and click **Finish**.

Delete a Business Process Schema Mapping

To delete a business process schema mapping:

1. From the **Deployment** menu, select **Web Services** > **Schema Mappings**.
2. In the BP Schema Mapping page, perform one of the following tasks:
   - Enter the name of the business process in the **Search** box and click **Go!**
   - Select **ALL** or the first letter in the name of the business process from **Alphabetically** drop-down list and click **Go!**
3. In the BP Schema Mappings page that is displayed, select the business process.

   The BP Mapping configuration is displayed in a pop-up window. Review the input schema and the output schema and verify that it is the correct business process schema mapping to delete.

   To close the pop-up window, click **Close**.
4. Click **Delete** next to the name of the business process.
5. When prompted with the following message, click **OK**.

   ```
   Are you sure you want to delete this BP Schema Mapping?
   ```
6. In the Confirm page that is displayed, click **Delete** to confirm and delete the BP Schema mapping.
7. Click **Return** to continue.

Create a UserName Security Token

To create a UserName security token:

1. From the Administration menu, select **Deployment** > **Web Services** > **Security Tokens**.
2. In the Security Tokens page, under **Create**, next to **Create Security Token**, click **Go!**
3. In the Create Security Token page that is displayed, enter information in the following fields and click **Next**.

Field	Description
Security Token Name	(Required) Enter a name for this token.
Token Description	(Required) Enter a description for this token.
Token Type	(Required) Select a token type from the drop-down list.

4. In the Create UserName Token page that is displayed, enter information in the following fields and click **Next**.

Field	Description
User Name	(Required) Enter a user name to be used with this token.
Password	(Required) Enter a password to be used with this token.
Digest	(Optional) Select the check box if you want the password to be digested instead of being stored as plain text. Digest passwords enable you to avoid storing or transmitting the password in a decipherable format.

5. In the Confirm page that is displayed, review the settings and perform one of the following tasks:
   - To change the UserName Security Token settings, click **Back**.
   - To abandon the UserName Security Token settings, click **Cancel**.
   - To confirm the UserName Security Token settings, click **Finish**.

6. Click **Return** to continue.

Edit a Security Token

To edit a security token:

1. From the Administration menu, select **Deployment** > **Web Services** > **Security Tokens**.
2. In the Security Tokens page, perform one of the following tasks:
   - Under Search, next to **Security Token Name**, type the name of the token you want to edit, click **Go!**
   - Under List, select **ALL** or the first letter in the name of the security tokens from the **Alphabetically** drop-down list, click **Go!**
   - Under List, select the security token type from the **Search By Token Type** drop-down list, click **Go!**
3. In the Security Token Management page that is displayed, click **source manager** next to the security token you want to edit.
4. Click **edit** next to the version you want to edit.
5. Edit the security token settings according to your requirements.
6. In the Security Token Versions page, select the security token under **OTHER Versions** section.
7. In the Confirm page, click **Finish**.
8. Click **Return** to continue.

Delete a Security Token

To delete a security token:

Note: It is recommended that you export and save a copy of the security token.

1. From the Administration menu, select **Deployment** > **Web Services** > **Security Tokens**.
2. In the Security Tokens page, perform one of the following tasks:
   - Under Search, next to **Security Token Name**, type the name of the token you want to delete, click **Go!**
   - Under List, select **ALL** or the first letter in the name of the security tokens from the **Alphabetically** drop-down list, click **Go!**
   - Under List, select the security token type from the **Search By Token Type** drop-down list, click **Go!**
3. In the Security Token Management page that is displayed, click **source manager** next to the security token you want to delete.
4. In the Security Token Source Manager page that is displayed, perform one of the following tasks:
   - Select the **Delete** check box for one or more of the token versions you want to delete, next to Delete Selected Versions, click **Go!**

     When prompted with the following message, click **OK** to delete or click **Cancel** to abort the operation.

     ```
     Are you sure you want to delete selected versions?
     ```
   - Next to **Delete All Versions**, click **Go!**

     When prompted with the following message, click **OK** to delete all versions or click **Cancel** to abort the operation.

     ```
     Are you sure you want to delete all versions of this token? It may be needed for internal processes. Prior to delete, it is recommended that you export and save a copy to offline storage.
     ```
5. In the Resource Summary page that is displayed, click **Next**.
6. In the Confirm page that is displayed, click **Delete**.
7. Click **Return** to continue.

Services Used By Web Services

The following table contains the names and descriptions of the services and adapters used to enable Web services for Sterling Integrator. 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.
SOA SSL Http Server Adapter	Configuration of the HTTP Server adapter specifically for Web services HTTPS transport. As delivered, it is set up to use the default perimeter server (node1 & local) and HTTP. **Note:** If you want to use an HTTPS certificate other than the default self-signed certificate, configure that change on the SSL configuration page of the adapter instance.
Dynamic Service Invoker Service	Invokes a service designated by the service parameter SVC_NAME. The primary business usage of this service is 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 Sterling Integrator-specific error information into standard Web services (SOAP) faults. The SOAP protocol requires specific fault handling mechanisms that allow Sterling Integrator-specific details to be added to the fault message. This service allows detailed error information for Sterling Integrator to be added to the appropriate SOAP fault message.

Service	Description
SOA Inbound	Processes inbound Web service request documents, separates the MIME parts, and validates signatures. Called by the system for a Web services consumer process for this application. **Note:** Configure and use the SOAInbound and SOAOutbound services if you want, as a consumer, to sign attachments or use signature processing based on XML signature. These features are not available with SOAInboundSecurityService and SOAOutboundSecurityService.
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. It also extracts the SOAP Envelope from the message and uploads it in the process data as a primary document.
SOAInboundSecurityService	Processes WS-Security headers (including timestamps, username token, signature verification, and decryption) in an incoming SOAP message. The input to this service is a SOAP document carrying WS-Security Header and the output is a valid/decrypted SOAP document. This service is compliant with WS-Security specification 1.0. **Note:** The ability to sign attachments or use signature processing is not available with SOAInboundSecurityService and SOAOutboundSecurityService. To obtain these features as a consumer, use the SOAInbound and SOAOutbound services.
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. **Note:** Configure and use the SOAInbound and SOAOutbound services if you want, as a consumer, to sign attachments or use signature processing based on XML signature. These features are not available with SOAInboundSecurityService and SOAOutboundSecurityService.
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 complement to the SOA Inbound Msg Processing Service.
SOAOutboundSecurityService	Creates a WS-Security Header that can carry security timestamps, username token, signature and encryption. The input to this service is a SOAP document without any WS-Security header and the output is a signed, encrypted

Service	Description
	SOAP document carrying a WS-Security header. This service with compliant with WS-Security specification 1.0. **Note:** The ability to sign attachments or use signature processing is not available with SOAInboundSecurityService and SOAOutboundSecurityService. To obtain these features as a consumer, use the SOAInbound and SOAOutbound services.
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.
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 carries a WS-RM header and is a WS-Reliable Message or not. Called by the system for Web service consumer processes.
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.

Associate an Inbound Message with a User ID

An inbound Web services message can be associated with a Sterling Integrator user ID and assigned permissions in order to avoid the potential security gap that occurs if all incoming messages are handled with administration rights.

This is done by using a business process override assignment syntax to add a message to a user mailbox.

1. Choose an instance of the Lightweight JDBC adapter.
2. Bootstrap to connect as specific user.

Example

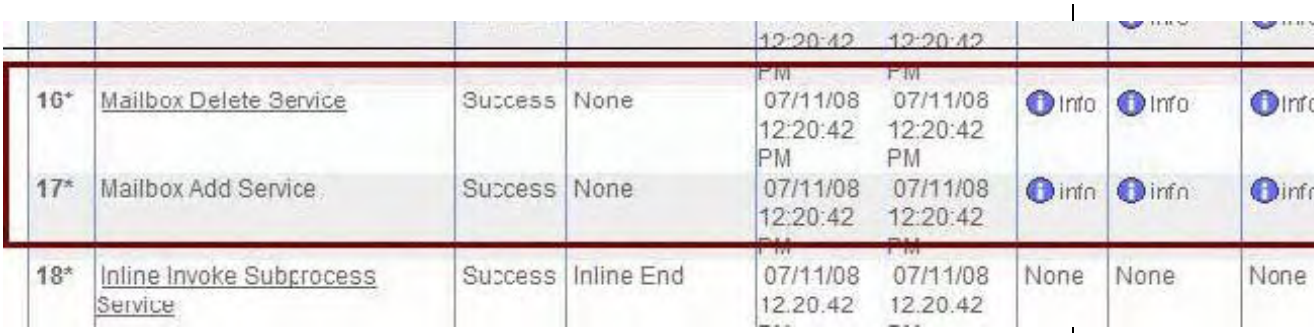
The following business process can be used in a Web service to do mailbox operations. It uses the setUserTokenService to add a user token to the workflow to use for mailbox operations. This is functionally equivalent to selecting a user in the user interface when you execute a business process.

To use this method, create a Web service and select a business process similar to the one below. Use any client, such as Axis, as the consumer to send the SOAP Request. The business process puts the primary document, the incoming SOAP Request, into a mailbox named mailbox1. A user called “mailboxAdmin” is also created with permission to access the “mailbox1” mailbox.

To send as an Inbound message attachment, make the attachment the Primary Document in the business process (non-.NET clients only).

As shown in the screenshot, the mailbox add succeeds in the WS-MessageHandler business process. Note that the business process as a whole is running with “None Available” as user, but that it executes the mailbox with user token as “mailboxAdmin.”

Note: Enabling the User Authentication in HTTP server adapter that runs Web services business process (SOA HTTP Server adapter) would be an alternative method.



16*	Mailbox Delete Service	Success	None	07/11/08 12:20:42 PM	07/11/08 12:20:42 PM	Info	Info	Info
17*	Mailbox Add Service	Success	None	07/11/08 12:20:42 PM	07/11/08 12:20:42 PM	Info	Info	Info
18*	Inline Invoke Subprocess Service	Success	Inline End	07/11/08 12:20:42	07/11/08 12:20:42	None	None	None

Exported XML for Business Process

```
<process name="default">
 <sequence>
  <!-- below line can be commented if security token is used-->
  <assign to="SECURITY_TOKEN_NAME">mailboxAdmin</assign>
  <assign to="USER_TOKEN" from="//SECURITY_TOKEN_NAME/text()"/>
  <!-- sets the user token in the BP -->
   <operation name="Set User Token Service">
    <participant name="SetUserTokenService"/>
    <output message="SetUserTokenServiceTypeInputMessage">
     <assign to="." from="*"></assign>
    </output>
    <input message="inmsg">
     <assign to="SetUserTokenServiceResults" from="*"></assign>
    </input>
   </operation>
   <operation name="Mailbox Delete Service">
    <participant name="MailboxDelete"/>
    <output message="MailboxDeleteServiceTypeInputMessage">
     <assign to="MailboxSelection">all</assign>
     <assign to="MessageNamePattern">message*</assign>
     <assign to="." from="*"></assign>
    </output>
    <input message="inmsg">
     <assign to="MailboxDeleteServiceResults" from="*"></assign>
    </input>
   </operation>
   <operation name="Mailbox Add Service">
    <participant name="MailboxAdd"/>
    <output message="MailboxAddServiceTypeInputMessage">
     <assign to="MailboxPath">/mailbox1</assign>
     <assign to="MessageName">message1</assign>
     <assign to="." from="*"></assign>
    </output>
    <input message="inmsg">
     <assign to="." from="*"></assign>
    </input>
   </operation>
 </sequence>
</process>
```

Exported Mailbox

Email from

```
<?xml version="1.0" encoding="UTF-8"?>
<APP_RESOURCES xmlns="http://www.stercomm.com/APP/APP_Resources"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
GISVersion="3000"
passphraseCheckString="rO0ABXQAB . . . 86lcgd">
<MBXMETA>
<MBXMETA>
<MAILBOX>
<BASEPATH>/mailbox1</BASEPATH>
<METADATA>
<PARENT_PATH></PARENT_PATH>
<PATH>/</PATH>
<DESCRIPTION>Root Mailbox</DESCRIPTION>
<PERMISSION>
<PERMISSION_ID>/.mbx</PERMISSION_ID>
<PERMISSION>/ Mailbox</PERMISSION>
<STATUS>1</STATUS>
<TYPE>1</TYPE>
</PERMISSION>
</METADATA>
<METADATA>
<PARENT_PATH>/</PARENT_PATH>
<PATH>/mailbox1</PATH>
<DESCRIPTION>new mailbox</DESCRIPTION>
<PERMISSION>
<PERMISSION_ID>/mailbox1.mbx</PERMISSION_ID>
<PERMISSION>/mailbox1 Mailbox</PERMISSION>
<STATUS>1</STATUS>
<TYPE>1</TYPE>
</PERMISSION>
<USERXPERMS>
<RECORD>
<USERXPERM>
<USER_ID>mailboxAdmin</USER_ID>
<PERMISSION_ID>/mailbox1.mbx</PERMISSION_ID>
<STATUS>1</STATUS>
<RIGHTS>2047</RIGHTS>
</USERXPERM>
<USER>
<USER_ID>mailboxAdmin</USER_ID>
<PASSWORD>naa4 . . . TIsDTtfo=</PASSWORD>
<LANG>en</LANG>
<EMAIL></EMAIL>
<FNAME>Mailbox</FNAME>
<LNAME>Admin</LNAME>
<PAGER></PAGER>
<VERSION></VERSION>
<DOWNLOAD_TIME></DOWNLOAD_TIME>
<SUPER>false</SUPER>
<PARENT_ID></PARENT_ID>
```

```
<ENTITY_ID>DEFAULT</ENTITY_ID>
<STATUS>1</STATUS>
<TIMEOUT>0</TIMEOUT>
<POLICY_ID></POLICY_ID>
<PWD_MOD_DATE>1215756571000</PWD_MOD_DATE>
<DASH_USER_ID>0</DASH_USER_ID>
<CONFIRM_VALUE></CONFIRM_VALUE>
<MODIFIED>1215756571000</MODIFIED>
<CREATED>1215756571000</CREATED>
<LAST_LOGIN></LAST_LOGIN>
<DISABLED></DISABLED>
<SECURITY_CODE>0</SECURITY_CODE>
<CHANGE_PASS_NEXT>false</CHANGE_PASS_NEXT>
</USER>
</RECORD>
</USERXPERMS>
</METADATA>
</MAILBOX>
</MBXMETA>
</MBXMETA>
<WEB_SERVICES>
<WEB_SERVICE>
<NAME>MailboxWebservice</NAME>
<DESCRIPTION>mailbox webservice to add SOAP messages to
mailbox</DESCRIPTION>
<NEW_SECURITY_SETTINGS>true</NEW_SECURITY_SETTINGS
><SYNC_BP_MODE>false</SYNC_BP_MODE>
<SOAP_TRANSPORT_BINDING>http</SOAP_TRANSPORT_BINDING>
<SECURITY>
<VERIFICATION_KEY></VERIFICATION_KEY>
<DECRYPTION_KEY></DECRYPTION_KEY>
<SECURITY_TOKEN_NAME></SECURITY_TOKEN_NAME>
<SECURITY_HEADER_ACTOR></SECURITY_HEADER_ACTOR>
<INSERT_MUSTUNDERSTAND>0</INSERT_MUSTUNDERSTAND>
<INSERT_TIME_STAMPS>0</INSERT_TIME_STAMPS>
<TIME_TO_LIVE>0</TIME_TO_LIVE>
<INSERT_USER_NAME_TOKEN>0</INSERT_USER_NAME_TOKEN>
<USER_NAME_TOKEN_NAME></USER_NAME_TOKEN_NAME>
<SIGNING_CERTIFICATE></SIGNING_CERTIFICATE>
<SIGNING_ALGO></SIGNING_ALGO>
<CANONICALIZATION_ALGO></CANONICALIZATION_ALGO>
<SIGNING_KEY_IDENTIFIER_TYPE>-1</SIGNING_KEY_IDENTIFIER_TYPE>
<ENCRYPTION_CERTIFICATE></ENCRYPTION_CERTIFICATE>
<SYMMETRIC_KEY_ALGO></SYMMETRIC_KEY_ALGO>
<KEY_ENCODING_ALGO></KEY_ENCODING_ALGO>
<ENCRYPT_KEY_IDENTIFIER_TYPE>0</ENCRYPT_KEY_IDENTIFIER_TYPE>
<SIGNATURE_ENCRYPTION_ORDER>-1</SIGNATURE_ENCRYPTION_ORDER></SECURITY>
<COMPONENTS>
<COMPONENT>
<COMPONENT_TYPE>BusinessProcess</COMPONENT_TYPE>
<COMPONENT_NAMES>
<COMPONENT_NAME>MailBoxAddBP</COMPONENT_NAME>
</COMPONENT_NAMES>
```

```
</COMPONENT>
</COMPONENTS>
<INLINE_ATTACHMENT>false</INLINE_ATTACHMENT>
<INPUT_ATTACHMENT>false</INPUT_ATTACHMENT>
<OUTPUT_ATTACHMENT>false</OUTPUT_ATTACHMENT>
<RELIABILITY_MODE>2</RELIABILITY_MODE>
<SOAP_CLAIM>-1</SOAP_CLAIM>
<WSDL_CLAIM>-1</WSDL_CLAIM>
<USERS></USERS>
</WEB_SERVICE>
</WEB_SERVICES>
<BPDEFS>
<BPDEF>
<LangResource>SIB64ENCOD . . . 9jZXNzPg0KDQo=</LangResource>
<ConfigResource>
<ConfDescription>v4</ConfDescription>
<ConfProcessName>MailBoxAddBP</ConfProcessName>
<ConfWFDID>231</ConfWFDID>
<ConfWFDVersion>4</ConfWFDVersion>
<OBJECT_VERSION>4</OBJECT_VERSION>
<SIResourceDefaultVersion>true</SIResourceDefaultVersion>
<ConfPersist>1</ConfPersist>
<ConfLifeSpan>-1</ConfLifeSpan>
<ConfRemoval>1</ConfRemoval>
<ConfDocStorage>4</ConfDocStorage>
<ConfPriority>4</ConfPriority>
<ConfRecoveryLevel>3</ConfRecoveryLevel>
<ConfOnfaultFlag>false</ConfOnfaultFlag>
<ConfStatus>1</ConfStatus>
<ConfLastUsed>admin</ConfLastUsed>
<ConfEncoding>None</ConfEncoding>
<ConfType>1</ConfType>
<ConfDocTracking>false</ConfDocTracking>
<ConfDeadLineInterval>-1</ConfDeadLineInterval>
<ConfFirstNotifyInterval>-1</ConfFirstNotifyInterval>
<ConfSecondNotifyInterval>-1</ConfSecondNotifyInterval>
<ConfEventLevel>2</ConfEventLevel>
<ConfCategory></ConfCategory>
</ConfigResource>
</BPDEF>
<BPDEF>
<LangResource>SIB64ENCODEPHByb2Nlc3MgbmFtZT0iTWFpbEJveEFkZEJQIj4NCiAgPHNlcXVlbmNlPg0
.
.
.
lcmF0aW9uPg0KICAgIA0KICA8L3NlcXVlbmNlPg0KPC9wcm9jZXNzPg0KDQo=</LangResource>
<ConfigResource>
<ConfDescription>v4</ConfDescription>
<ConfProcessName>MailBoxAddBP</ConfProcessName>
<ConfWFDID>231</ConfWFDID>
<ConfWFDVersion>4</ConfWFDVersion>
<OBJECT_VERSION>4</OBJECT_VERSION>
<SIResourceDefaultVersion>true</SIResourceDefaultVersion>
```

```
<ConfPersist>1</ConfPersist>
<ConfLifeSpan>-1</ConfLifeSpan>
<ConfRemoval>1</ConfRemoval>
<ConfDocStorage>4</ConfDocStorage>
<ConfPriority>4</ConfPriority>
<ConfRecoveryLevel>3</ConfRecoveryLevel>
<ConfOnfaultFlag>false</ConfOnfaultFlag>
<ConfStatus>1</ConfStatus>
<ConfLastUsed>admin</ConfLastUsed>
<ConfEncoding>None</ConfEncoding>
<ConfType>1</ConfType>
<ConfDocTracking>false</ConfDocTracking>
<ConfDeadLineInterval>-1</ConfDeadLineInterval>
<ConfFirstNotifyInterval>-1</ConfFirstNotifyInterval>
<ConfSecondNotifyInterval>-1</ConfSecondNotifyInterval>
<ConfEventLevel>2</ConfEventLevel>
<ConfCategory></ConfCategory>
</ConfigResource>
</BPDEF>
</BPDEFS>
<USERS>
<USER>
<METADATA>
<USER_ID>mailboxAdmin</USER_ID>
<PASSWORD>naa. . . tfo=</PASSWORD>
<SALT>Wvxhzf75vMtaSgjYtaW4Sg==</SALT>
<LANG>en</LANG>
<EMAIL></EMAIL>
<FNAME>Mailbox</FNAME>
<LNAME>Admin</LNAME>
<PAGER></PAGER>
<VERSION></VERSION>
<DOWNLOAD_TIME></DOWNLOAD_TIME>
<SUPER>false</SUPER>
<PARENT_ID></PARENT_ID>
<ENTITY_ID>Hub Organization</ENTITY_ID>
<STATUS>1</STATUS>
<TIMEOUT>0</TIMEOUT>
<POLICY_ID></POLICY_ID>
<PWD_MOD_DATE>1215756571000</PWD_MOD_DATE>
<DASH_USER_ID>0</DASH_USER_ID>
<CONFIRM_VALUE></CONFIRM_VALUE>
<MODIFIED>1215756571000</MODIFIED>
<CREATED>1215756571000</CREATED>
<LAST_LOGIN></LAST_LOGIN
><DISABLED></DISABLED>
<SECURITY_CODE>0</SECURITY_CODE>
<CHANGE_PASS_NEXT>false</CHANGE_PASS_NEXT>
</METADATA>
<USERDEPENDENTS>
<USERXPERMS>
<USERXPERM>
<METADATA>
```

```
<USER_ID>mailboxAdmin</USER_ID>
<PERMISSION_ID>/mailbox1.mbx</PERMISSION_ID>
<STATUS>1</STATUS>
<RIGHTS>2047</RIGHTS>
</METADATA>
</USERXPERM>
</USERXPERMS>
<PERMISSIONS>
<PERMISSION>
<METADATA>
<PERMISSION_ID>/mailbox1.mbx</PERMISSION_ID>
<PERMISSION>/mailbox1Mailbox</PERMISSION>
<STATUS>1</STATUS>
<TYPE>1</TYPE></METADATA>
</PERMISSION>
</PERMISSIONS>
<USERXGROUPS>
<USERXGROUP>
<METADATA>
<GROUP_ID>mbiusers</GROUP_ID>
<USER_SUB_GROUP_ID>mailboxAdmin</USER_SUB_GROUP_ID>
</METADATA>
</USERXGROUP>
<USERXGROUP>
<METADATA>
<GROUP_ID>mboxadmins</GROUP_ID>
<USER_SUB_GROUP_ID>mailboxAdmin</USER_SUB_GROUP_ID>
</METADATA>
</USERXGROUP>
<USERXGROUP>
<METADATA>
<GROUP_ID>MAILBOX</GROUP_ID>
<USER_SUB_GROUP_ID>mailboxAdmin</USER_SUB_GROUP_ID>
</METADATA>
</USERXGROUP>
</USERXGROUPS>
<GROUPS>
<GROUP>
<METADATA>
<GROUP_ID>mbiusers</GROUP_ID>
<GROUP_NAME>Mailbox Browser Interface Users</GROUP_NAME>
<ENTITY_ID></ENTITY_ID>
<STATUS>1</STATUS>
<GROUP_OWNER_ID></GROUP_OWNER_ID>
</METADATA>
<GROUPDEPENDENTS>
<GROUPXPERMS>
<GROUPXPERM>
<METADATA>
<GROUP_ID>mbiusers</GROUP_ID>
```

Business Processes Used By Web Services

The following table lists the business processes provided along with Sterling Integrator for use with Web services, including a description and related services.

Business Process (.bpml)	Description	Related Services
WS_MessageHandler	Uses a dynamic service created in Sterling Integrator to send a SOAP message through HTTP to an endpoint configured in Sterling Integrator itself (where Sterling Integrator is the consumer as well as provider).	SOA Inbound Message Processing Service
WS_MessageHandler_SSL	Same as WS_MessageHandler, with SSL enabled.	
WS_MessageHandler_Sync	Same as WS_MessageHandler, in synchronous mode.	
WS_MessageHandler_Sync_SSL	Same as WS_MessageHandler, in synchronous mode with SSL enabled.	
WS_RequestHandler	Uses a dynamic service created in Sterling Integrator to send a SOAP message through HTTP to an endpoint configured in Sterling Integrator itself (where Sterling Integrator is the consumer as well as provider).	• SOA Outbound Message Processing Service • SOA Inbound Security Service • SOA Outbound Security Service
WS_RequestHandler_Sync	Same as WS_RequestHandler, in synchronous mode	

Configuration Overrides for Web Services

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

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 the /*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 Sterling Integrator, 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` command is run. Always make changes to the template file, `soa.properties.in`, to maintain the changes.

To edit the soa.properties.in file:

1. In the /*install_dir*/properties (*install_dir*\properties for Windows) directory, locate the `soa.properties.in` file.
2. Open the file using a text editor and modify the file based on your requirements.

   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 users:

Property	Description
useCache=true	Indicates that generated WSDL will be cached by Sterling Integrator to speed up retrieval.
class.wsdl=com.sterlingcomerce.server.services.soa.util.WSDLServiceInfo	Class used by Service Info service to generate WSDL. **Note:** Do not change this property.
class.xml=com.sterlingcomerce.server.services.soa.util.XMLServiceInfo	Class used by Service Info service to generate raw XML representation of services.

Property	Description
	Note: Do not change this property.
wsdlMessage=/wsdl:definitions/wsdl:message	XPath entry of the message element within a WSDL which will be required during insertion of the <wsi:Claim> element within the message element. The presence of this element indicates the WS-I conformity of the message node in the generated WSDL. **Note:** Do not change the value of this property. Removal of this property indicates that the message node in the WSDL generated by Sterling Integrator is not WS-I compliant.
wsdlportType=/wsdl:definitions/wsdl:portType	XPath entry of the portType element within a WSDL which will be required during insertion of the <wsi:Claim> element within the portType element. The presence of this element indicates the WS-I conformity of the portType node in the generated WSDL. **Note:** Do not change the value of this property. Removal of this property indicates that the portType node in the WSDL generated by Sterling Integrator is not WS-I compliant.
wsdlBinding=/wsdl:definitions/wsdl:binding	The XPath entry of the binding element within a WSDL which will be required during insertion of the <wsi:Claim> element within the binding element. The presence of this element indicates the WS-I conformity of the binding node in the generated WSDL. **Note:** Do not change the value of this property. Removal of this property indicates that the binding node in the WSDL generated by Sterling Integrator is not WS-I compliant.
wsdlOperation=/wsdl:definitions/wsdl:portType/wsdl:operation	The XPath entry of the operation element within a WSDL which will be required during insertion of the <wsi:Claim> element within the operation element. The presence of this element indicates the WS-I conformity of the operation node in the generated WSDL.

Property	Description
	Note: Do not change the value of this property. Removal of this property indicates that the operation node in the WSDL generated by Sterling Integrator is not WS-I compliant.
defaultBaseURL=http://&HOST_NAME;:&SOA_PORT;	Default base URL for accessing Web services. Additional HTTP Server adapters can be configured. **Note:** 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. **Note:** 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. **Note:** Do not change this property.
defaultSOAPort=&SOA_PORT;	Default SOA port for accessing Web services. Additional HTTP Server adapters can be configured. **Note:** Do not change this property.
attachmentMimeType=application/octetstream	MIME format used for SOAP messages with attachments.
signatureRequired=false	Override to force signature usage.
signatureTrigger=/xmldsig	Indicator used to determine if a given message is signed. **Note:** Do not change this property.
signatureMaxScan=8192	Tuning parameter for signature determination. **Note:** Do not change this property.
enforceStrongTyping=false	When this parameter is set to false, all parameters are made optional. This is used when internal service definition is inconsistent with actual usage.

Property	Description
	Prevents marking of multiple required fields as required.
In addition to the previous parameters, the following overrides enable you to modify the WSDL. These parameters operate at a Web services configuration level, which allows for more flexibility: **Note:** Exercise caution when modifying 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 this parameter is 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 this parameter is 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.

3. Run the `setupfiles.sh` (`setupfiles.cmd` for Windows) file to apply the changes.

SOAP Fault Messages

The following table lists the SOAP Fault messages generated by the system services 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.

Provider WSDL Example

The following is an 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 is inserted into the “types” section of the input message in the WSDL generated. The `mesaAuth` element is a mechanism internal to Sterling Integrator, and the resulting WSDL is not compliant with WS-I Basic Profile 1.1. If you do not select a consumer, the `mesaAuth` element is not inserted, and the resulting WSDL is compliant with WS-I Basic Profile 1.1.

Input XML Schema for Web Services

The following is an example of an XML input schema (ExpenseReportInput.xsd):

```
<?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 XML Schema for Web Services

The following is an example of an XML output schema (ExpenseReportOutput.xsd):

```
<?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 Example: Enabling Customer Business

This is an example of a Web service designed to enable customer 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, for example, how many shifts to schedule for the next two weeks. 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 service called `Inventory_Check` with the following related components:

Component	Name
Business Process	Bead_Inquiry.bp
Service	Lightweight JDBC adapter configuration named “Bead_LTWTJDBC” (used in the business process)
User Account	One new Sterling Integrator 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 service, `Inventory_Check`, and then adds the business process and the user account to it.

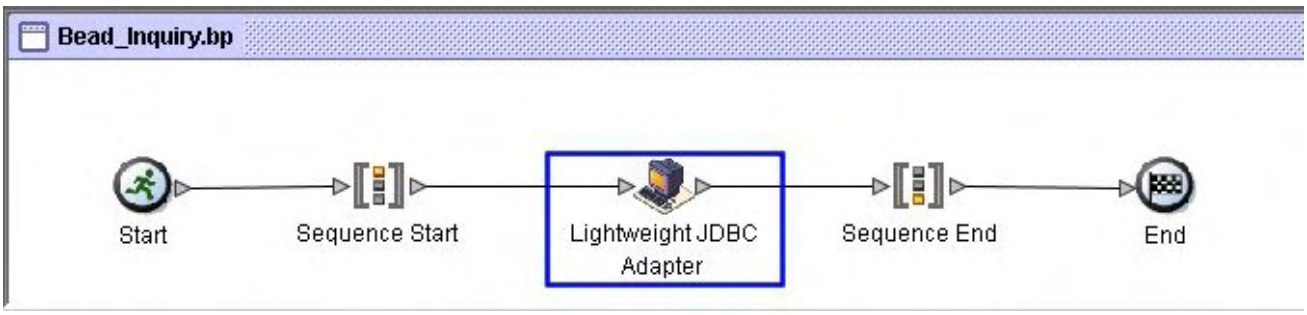
   Note: It is not necessary to add the Lightweight JDBC adapter configuration, Bead_LTWTJDBC, to the Web service because it is part of the business process that is being added to the Web services.

3. Company A generates the WSDL for the Web service 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 sent 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

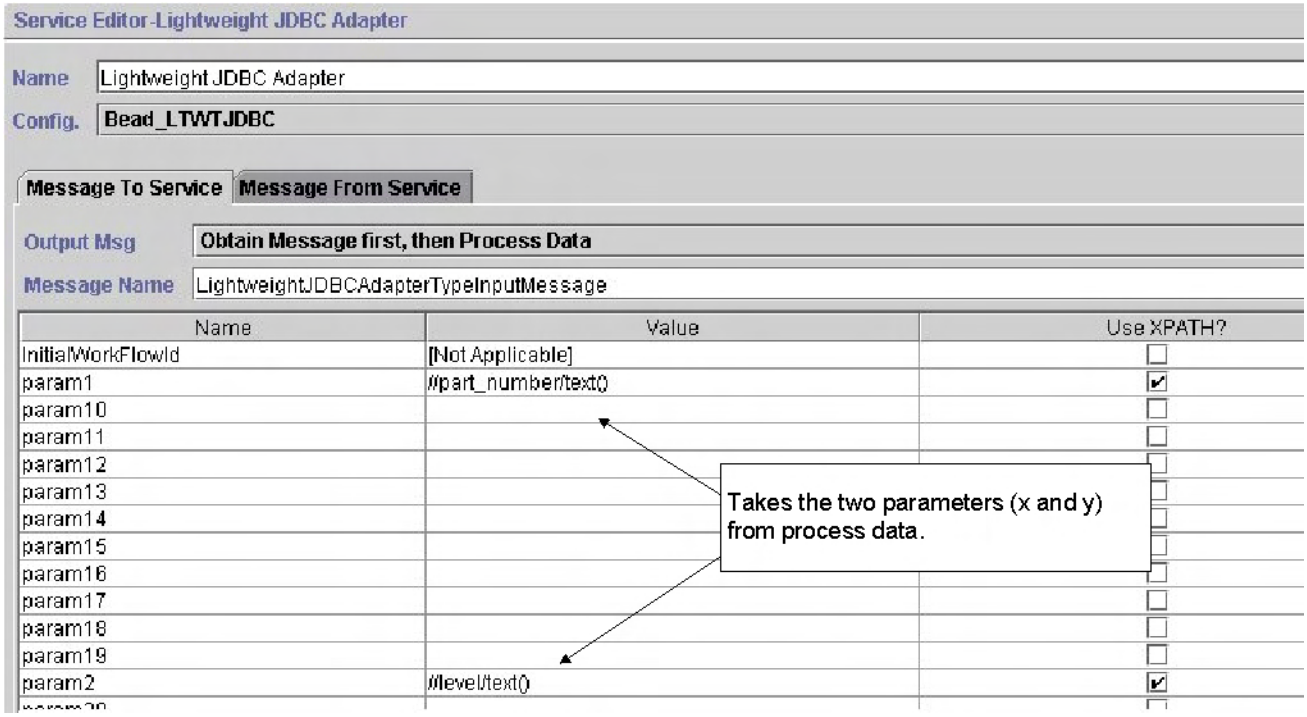
The following figure shows how the business process might look in the GPM:



The following list and the figures show the GPM parameters for the business process:

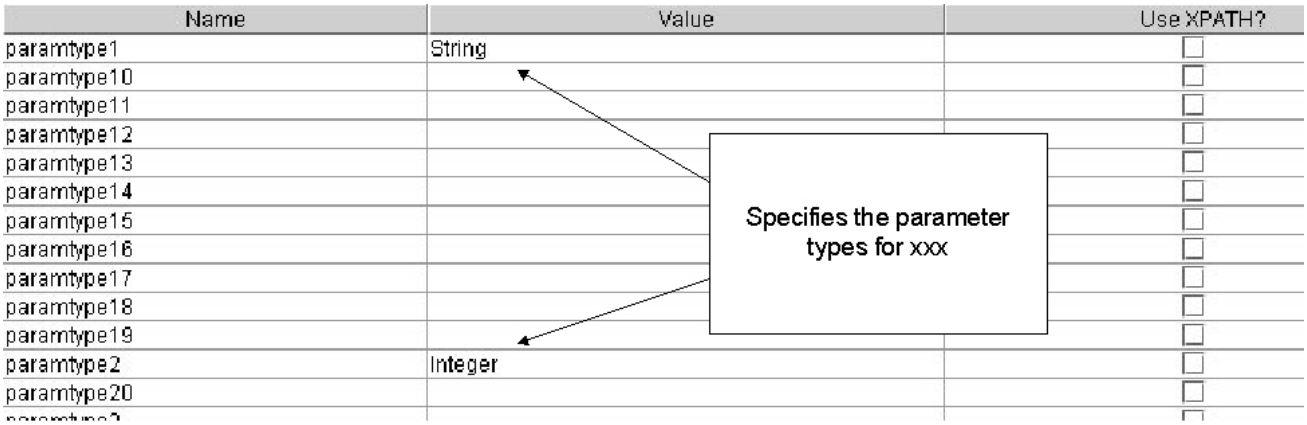
• **GPM Parameters, Part 1**

The following figure shows the GPM parameters for the business process:



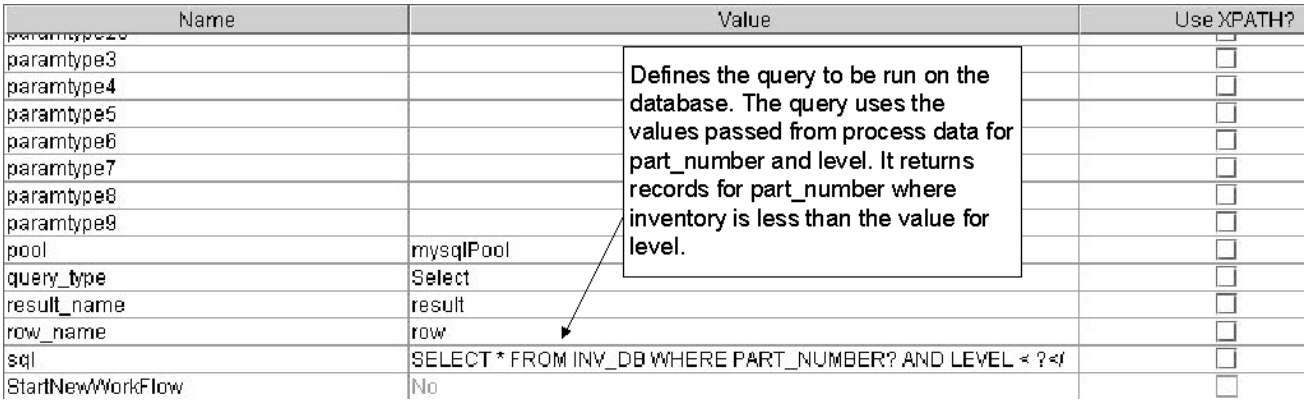
• **GPM Parameters, Part 2 (continued)**

The following figure shows the GPM parameters for the business process:



• **GPM Parameters, Part 3 (continued)**

The following figure shows the GPM parameters for the business process:



Bead_InQuiry Business Process Source

The following BPML is the source for the Bead_InQuiry business process 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 Service

The following example shows the WSDL for the Inventory_Check Web service:

```
<?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>
```

Web Services Consumer Example: Stock Quote Service

In this example, a consumer submits a stock symbol and receives a stock price from a Web service in Sterling Integrator.

This requires the following tasks:

- Create a Stock Quote Service in Sterling Integrator. For more information, refer to the topic *Create a Stock Quote Web Service* on page 89.
- Create a Web Services Consumer for the Stock Quote Service. For more information, refer to the topic *Create a Web Services Consumer for the Stock Quote Service* on page 93.

Create a Stock Quote Web Service

To create a Stock Quote Web service in Sterling Integrator:

1. Create an input schema that takes a stock symbol as input by placing the following schema in a file named `stockQuoteInput.xsd`:

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema elementFormDefault="qualified"
targetNamespace="http://www.webservice.stockquotein"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="GetQuote">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="1" name="symbol" type="xs:string"
 />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

   For this input schema, Sterling Integrator expects a SOAP Request similar to the following:

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
```

```
  <soap:Body>
    <GetQuote xmlns=" http://www.webservice.stockquotein">
      <symbol>string</symbol>
    </GetQuote>
  </soap:Body>
</soap:Envelope>
```

2. Create an output schema that gives the output as price by placing the following schema in a file named `stockQuoteOutput.xsd`:

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema elementFormDefault="qualified"
targetNamespace="http://www.webservice.stockquoteout"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="GetQuoteResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="1" name="GetQuoteResult"
type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

   The SOAP response sent from Sterling Integrator using this output schema looks like the following:

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetQuoteResponse xmlns="http://www.webservice.stockquoteout">
    <GetQuoteResult>string</GetQuoteResult>
    </GetQuoteResponse>
  </soap:Body>
</soap:Envelope>
```

3. From the **Administration** menu, select **Deployment** > **Resource Manager** > **Import/Export** and import the schemas into Sterling Integrator.
4. From the **Administration** menu, select **Deployment** >**Schemas** and check in these schemas.

   Do not modify the target Namespace.

5. Add the following to `namespaces.properties` file to create the namespaces for use in the input and output schemas.

```
stockout=http://www.webservice.stockquoteout
stockin=http://www.webservice.stockquotein
```

6. Create a simple business process such as the following in Sterling Integrator to handle the stock quote service functionality. It must return a price as the response by creating the output XML and assigning it to WebservicesResponseNode.

```
<process name="StockQuoteBP">
  <sequence name="AssignSequence">
  <assign to="stockout:GetQuoteResult">120.35</assign>
  <assign to="stockout:GetQuoteResponse"
```

```
   from="//*[namespace-uri()='http://www.webservice.stockquoteout'
   and local-name()='GetQuoteResult']"/>
  <assign to="WebservicesResponseNode"
   from="//*[namespace-uri()='http://www.webservice.stockquoteout'
   and local-name()='GetQuoteResponse']"/>
  </sequence>
</process>
```

Check in the business process into Sterling Integrator as StockQuoteBP.

7. From the Administration menu, select **Deployment** > **Web Services** > **Schema Mappings**, create a new business process schema mapping.
   a) Select `StockQuoteBP` from the Business Process drop-down list.
   b) Select `stockQuoteInput.xsd` from the Input Schema drop-down list.
   c) Select `stockQuoteOutput.xsd` from the Output Schema drop-down list.
   d) Select the check box next to `Validate with Input Schema` and `Validate with Output Schema`.
8. Create a Web service, `StockQuoteWebservice`, in Sterling Integrator that includes this business process.
   - Clear the `Use Legacy Settings` check box.
   - Assign `StockQuoteBP` as the business process.
   - Use default settings in all other pages.
9. Generate the WSDL for the Web service as shown in the following example:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<wsdl:definitions name=”StockQuoteWebservice”
  targetNamespace=”http://www.sterlingcommerce.com/mesa”
  xmlns:mesa=”http://www.sterlingcommerce.com/mesa”
  xmlns:mesa_xsd=”http://www.sterlingcommerce.com/mesa”
  xmlns:mime=”http://schemas.xmlsoap.org/wsdl/mime/”
  xmlns:soap=”http://schemas.xmlsoap.org/wsdl/soap/”
  xmlns:tns0=”http://www.webservice.stockquotein”
  xmlns:tns1=”http://www.webservice.stockquoteout”
  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”
      xmlns=”http://www.sterlingcommerce.com/mesa”
      xmlns:tns=”http://www.sterlingcommerce.com/mesa”
    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:schema>
    <xs:schema elementFormDefault=”qualified”
      targetNamespace=”http://www.webservice.stockquotein”
      xmlns=”http://www.webservice.stockquotein”
      xmlns:tns0=”http://www.webservice.stockquotein”
    xmlns:xs=”http://www.w3.org/2001/XMLSchema”>
    <xs:element name=”GetQuote”>
      <xs:complexType>
        <xs:sequence>
          <xs:element maxOccurs=”1” minOccurs=”0” name=”symbol”
type=”xs:string”/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:schema>
  <xs:schema elementFormDefault=”qualified”
    targetNamespace=”http://www.webservice.stockquoteout”
    xmlns=”http://www.webservice.stockquoteout”
    xmlns:tns1=”http://www.webservice.stockquoteout”
  xmlns:xs=”http://www.w3.org/2001/XMLSchema”>
     <xs:element name=”GetQuoteResponse”>
       <xs:complexType>
         <xs:sequence>
           <xs:element maxOccurs=”1” minOccurs=”0” name=”GetQuoteResult”
type=”xs:string”/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:schema>
  </wsdl:types>
  <wsdl:message name=”MESAResponse”>
   <wsdl:part element=”mesa:processData” name=”parameters”/>
  </wsdl:message>
  <wsdl:message name=”StockQuoteBP”>
    <wsdl:part element=”tns0:GetQuote” name=”parameters”/>
```

```
  </wsdl:message>
  <wsdl:message name="MESAFault">
    <wsdl:part element="mesa:mesaFault" name="parameters"/>
  </wsdl:message>
  <wsdl:message name="GISGeneric">
    <wsdl:part element="mesa:processData" name="parameters"/>
  </wsdl:message>
  <wsdl:message name="StockQuoteBPOutput">
    <wsdl:part element="tns1:GetQuoteResponse" name="parameters"/>
  </wsdl:message>
  <wsdl:portType name="GISPortType">
    <wsdl:operation name="executeStockQuoteBP">
      <wsdl:input message="mesa:StockQuoteBP"/>
      <wsdl:output message="mesa:StockQuoteBPOutput"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="GISBinding" type="mesa:GISPortType">
    <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="executeStockQuoteBP">
      <soap:operation soapAction="sii:StockQuoteBP" style="document"/>
        <wsdl:input>
          <soap:body parts="parameters" use="literal"/>
        </wsdl:input>
        <wsdl:output>
          <soap:body use="literal"/>
        </wsdl:output>
      </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="StockQuoteWebservice">
      <wsdl:port binding="mesa:GISBinding" name="GISPort">
        <soap:address
location="http://10.30.2.249:46940/soap-new?service=StockQuoteWebservice"/>
      </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

10. Check in the WSDL and enable it for business processes.
11. Export the WSDL to a location accessible to the consumer, or send an XML file.

Create a Web Services Consumer for the Stock Quote Service

To create a Web services consumer for the Stock Quote service:

1. Check-in the WSDL.

   For more information, refer to the topic *Check In a WSDL for a Web Service* on page 45.

2. Create a business process named `StockQuoteConsumer.bp`:
   a) Create an entry in the GPM pallet corresponding to the checked-in WSDL, which can be used while writing the business process.

b) Use the following “assign” statements and “DomToDoc” service to create a primary document for the execution of the business process:

```
<assign to="stockin:symbol">GISSC</assign>
<assign to="stockin:GetQuote"
from="//*[namespace-uri()='http://www.webservice.stockquotein'
and local-name()='symbol']"/>
<assign to="Root"
from="//*[namespace-uri()='http://www.webservice.stockquotein' and
local-name()='GetQuote']"/>
<assign name="Assign" to="."
from="DOMToDoc(/ProcessData/Root,&amp;apos;PrimaryDocument&amp;apos;)"></assign>
```

c) Add the following lines in the `namespaces.properties` file:

```
stockout=http://www.webservice.stockquoteout
```

```
stockin=http://www.webservice.stockquotein
```

The complete business process (including the input document) is as follows:

```
<process name="default">
  <sequence>
    <assign to="stockin:symbol">GISSC</assign>
    <assign to="stockin:GetQuote"
from="//*[namespace-uri()='http://www.webservice.stockquotein'
      and local-name()='symbol']"/>
    <assign to="Root"
from="//*[namespace-uri()='http://www.webservice.stockquotein' and
      local-name()='GetQuote']"/>
    <assign name="Assign" to="."
from="DOMToDoc(/ProcessData/Root,&amp;apos;PrimaryDocument&amp;apos;)">
      </assign>
    <assign name="Assign" to="GetQuote"
from="//PrimaryDocument/@SCIObjectID"></assign>
    <operation name="executeStockQuoteBP">
      <participant name="DS_WS_LIVEQUOTESERVICE_GIS_PORT1_OPE1_Instance"/>
      <output message="DS_WS_LIVEQUOTESERVICE_GIS_PORT1_OPE1InputMessage">
        <assign to="." from="*"></assign>
      </output>
      <input message="inmsg">
        <assign to="." from="*"></assign>
      </input>
    </operation>

    <operation name="SOA Outbound Message Processing Service">
      <participant name="SOAOutboundMsgProcessingService_Instance"/>
      <output message="SOAOutboundMsgProcessingTypeInputMessage">
        <assign to="." from="*"></assign>
      </output>
      <input message="inmsg">
        <assign to="." from="*"></assign>
      </input>
    </operation>
    <operation name="HTTP Client Begin Session Service">
      <participant name="HTTPClientBeginSession"/>
```

```
      <output message="HTTPClientBeginSessionServiceTypeInputMessage">
        <assign
to="HTTPClientAdapter">HttpClientAdapter_DynamicService</assign>
        <assign to="RemoteHost">10.11.23.31</assign>
        <assign to="RemotePort">30040</assign>
        <assign to="." from="*"></assign>
      </output>
      <input message="inmsg">
        <assign to="." from="*"></assign>
      </input>
    </operation>

    <operation name="HTTP Client POST Service">
      <participant name="HTTPClientPost"/>
      <output message="HTTPClientPostServiceTypeInputMessage">
        <assign to="RawRequest">true</assign>
        <assign to="RawResponse">true</assign>
        <assign to="URI">/soap-new?service=StockQuoteWebservice</assign>
        <assign to="." from="*"></assign>
      </output>
      <input message="inmsg">
        <assign to="." from="*"></assign>
      </input>
    </operation>

    <operation name="HTTP Client End Session Service">
      <participant name="HTTPClientEndSession"/>
      <output message="HTTPClientEndSessionServiceTypeInputMessage">
        <assign to="." from="*"></assign>
      </output>
      <input message="inmsg">
        <assign to="." from="*"></assign>
      </input>
    </operation>

    <operation name="SOA Inbound Message Processing Service">
      <participant name="SOAInboundMsgProcessingService_Instance"/>
      <output message="SOAInboundMsgProcessingServiceInputMessage">
        <assign to="." from="*"></assign>
      </output>
      <input message="inmsg">
        <assign to="." from="*"></assign>
      </input>
    </operation>
  </sequence>
</process>
```

3. Execute the business process.
   This will be the response in the primary document:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header>
```

```
    <wsse:Security
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-

      secext-1.0.xsd"/>
  </soapenv:Header>
  <soapenv:Body>
    <stockout:GetQuoteResponse
xmlns:stockout="http://www.webservice.stockquoteout">
    <stockout:GetQuoteResult>120.35</stockout:GetQuoteResult>
    </stockout:GetQuoteResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Error Messages

If you see the following Integer error message, use `long` when configuring the LWJDBC adapter:

```
Status ReportName: WS_MessageHandler Instance ID:11812
Service Name: Lightweight_JDBC_Insert java.lang.NumberFormatException:
For input string: "6148888888"
```

Dynamically Add, Remove, or Modify an XML Namespace for Web Services

Sterling Integrator has the ability to dynamically add, remove, or modify an XML namespace for Web Services without restarting Sterling Integrator.

When Sterling Integrator gets a request for a namespace that it doesn't recognize, it reloads the namespace properties, the customer_overrides properties, and all the extended properties files and checks for new namespaces. Sterling Integrator will store the new namespace in the cache for future use. Additionally, you can now refresh the namespaces in the properties files using a new OPS command that allows you the ability to add, modify, or remove namespaces in the properties files, you can also change these properties in the customer_overrides.properties file.

Adding a Namespace

Complete these steps to add a new namespace:

1. Create a business process similar to the example below:

```
<process name="namespaces_test_add">
<sequence>
<assign to="temp/@Algorithm" from="'http://www.w3.org/2000/09/xmldsig#test'"/>
<assign to="ds:Transforms/ds:Transform" from="temp/@*"/>
<assign to="ds1:Transforms/ds1:Transform" from="temp/@*"/>
</sequence>
</process>
```

2. Add a new namespace similar to the example below in the namespaces.properties file or in a new extended properties file.

   Note: You can add a namespace to namespace.properties or namespace.properties_*_ext or customer_overrides.properties

```
ds1 = http://www.w3.org/2000/09/xmldsig_ds1#
```

3. Run the assigned business process. *namespaces_test_add* business process.

4. The following will appear in the Process Data if the business process is running successfully:

```
<ds1:Transforms xmlns:ds1="http://www.w3.org/2000/09/xmldsig_ds1#">
<ds1:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#test"/>
</ds1:Transforms>
```

Modifying a Namespace

Complete these steps to update an existing namespace:

1. Create a business process similar to the one below:

```
<process name="namespaces_test_update">
<sequence>
<assign to="temp/@Algorithm" from="'http://www.w3.org/2000/09/xmldsig#test'"/>

<assign to="ds:Transforms/ds:Transform" from="temp/@*"/>
</sequence>
</process>
```

2. Update an existing namespace similar to the following example in the namespaces.properties file.

```
ds = http://www.w3.org/2000/09/xmldsig_update#
```

3. From the install root directory, run the OPS command:

```
./bin/opscmd.sh -cREFRESHNAMESPACES -nnode1
```

4. Run the assigned business process.
5. If the business process runs sucessfully the following The following will appear in the Process Data if the business process is running successfully.

```
<temp Algorithm="http://www.w3.org/2000/09/xmldsig#test"/>
<ds:Transforms xmlns:ds="http://www.w3.org/2000/09/xmldsig_update#">
<ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#test"/>
</ds:Transforms>
```

Removing a Namespace

Complete these steps to remove an existing namespace:

1. Create a business process similar to the following example:

```
<process name="namespaces_test_remove">
<sequence>
<assign to="temp/@Algorithm" from="'http://www.w3.org/2000/09/xmldsig#test'"/>

<assign to="ds:Transforms/ds:Transform" from="temp/@*"/>
</sequence>
</process>
```

2. Remove an existing namespace similar to the following example in the namespaces.properties file.
   ```
   ds = http://www.w3.org/2000/09/xmldsig_update#
   ```
3. From the install root directory, run the OPS command:
   ```
   ./bin/opscmd.sh -cREFRESHNAMESPACES -nnode1
   ```
   The cache clears.
4. Run the assigned business process.
5. The business process fails because it cannot refer to the “ds” namespace.

Frequently Asked Questions

The following table lists the frequently asked questions about Web Services Interoperability:

Question	Answer
Does Xfire-generated code (using wsgen) work with Sterling Integrator as a provider?	Yes, it does. Enable Use New Settings while configuring the Web service. **Note:** A namespace stripping issue that prevented the use of Xfire-generated code was addressed in Gentran Integration Suite 4.3.
Why does Apache Axis give a `Signature verification failed` error for the signed message sent from Sterling Integrator?	This is a known issue in Axis, discussed in many forums. Possible reasons for failure include: • CanonicalizationAlgorithm used is not exclusive type. • Handler configuration used by Axis for verifying the signature is not proper. (Only one handler should be used for the incoming soap message to the Axis engine.)

When attachments are enabled in a Sterling Integrator Web service, the (Axis 1.x) wsdl2java-generated code sends a message to Sterling Integrator that appears to be RPC style rather than the expected doc/lit style. Why?	Axis treats the WSDL with attachment settings as RPC-style WSDL, even though the WSDL clearly says that it is doc/lit style. As a workaround, add the following two lines to the GISBindingStub class generated (in the method _initOperationDescXX()
	`oper.setStyle(org.apache.axis.constants.Style.DOCUMENT);`
	`oper.setUse(org.apache.axis.constants.Use.LITERAL);`

Xfire throws an exception, `could not unmarshall type null` when DOMInHandler is added for WS-Security processing.	This is due to a bug in Xfire. Because Xfire is open source, you can obtain the code and modify the org.codehaus.xfire.util.stax.W3CDOMStreamReader class
	Change the method to StaxType as follows:
	`// import org.jdom.Attribute;`
	`public static String toStaxType(short jdom) {`
	`switch (jdom) { case Attribute.CDATA_TYPE:return "CDATA"; case Attribute.ID_TYPE:return "ID";`
	`case Attribute.IDREF_TYPE:return "IDREF"; case Attribute.IDREFS_TYPE:return "IDREFS";`
	`case Attribute.ENTITY_TYPE:return "ENTITY"; case Attribute.ENTITIES_TYPE:return`

Question	Answer
	`"ENTITIES"; case Attribute.ENUMERATED_TYPE:return "ENUMERATED"; case Attribute.NMTOKEN_TYPE:return "NMTOKEN"; case Attribute.NMTOKENS_TYPE:return "NMTOKENS"; case Attribute.NOTATION_TYPE:return "NOTATION"; default:return null; } }`
Why things do not work when I select Subject key identifier as the Key identifier type for encryption or signature?	Your Certificate may not have the required Subject key identifier extensions. These need to be added to the certificate by the CA (Certification Authority) or, if the certificate is self-signed, by you. Sterling Commerce recommends using openSSL instead of using Sterling Integrator or keytool to generate a self-signed certificate in this case, since OpenSSL allows you to generate certificates with extensions (in this case, SKI). Modify the `openssl.cnf` file to generate a certificate of this type.
Axis throws the message, `Invalid timestamp The security semantics of message have expired` even though the timestamp interval, as provided in the Sterling Integrator Provider Response Security Settings, is valid.	This is a problem on the Axis side. The maximum time, for which a SOAP message with a timestamp is considered valid by Axis, is five minutes. Any message processed after five minutes is considered invalid and the above error is thrown. This situation can arise if the machines on which consumer and provider are deployed have a time/date difference of more than five minutes.
Axis2, Java Enterprise Edition (EE) Web Services Developers Pack does not generate stubs (proxies) with our WSDL if consumers are selected or the inline attachment is selected.	Selecting Consumers or inline attachment adds multiple "parts" to the input "message" in the WSDL. This is not allowed by WS-I Basic profile specification, which specifies: • R2201 A document-literal binding in a DESCRIPTION MUST, in each of its soapbind:body element(s), have at most one part listed in the parts attribute, if the parts attribute is specified. • R2210 If a document-literal binding in a DESCRIPTION does not specify the parts attribute on a soapbind:body element, the corresponding abstract wsdl:message MUST define zero or one wsdl:parts. This means that, if we select consumers or inline attachment, the WSDL does not conform to the WS-I Basic Profile specification and hence is not parsed properly in Axis2 and the Java EE Web Services Developer Pack.
Java EE Web services that use the XWS-Security module for WS-Security do not work with our Web services provider if the key identifier for encryption/signature is X509KeyIdentifier.	This is because the Java EE XWS-Security module does not support X509KeyIdentifier. It supports only DirectReference, IssuerSerial, and SubjectKeyIdentifier. You can use one of these when configuring Web services in Sterling Integrator.

Question	Answer

The different Web service modules are having problems with issuer serial information provided by our Web services while decrypting the data when IssuerSerial is selected as key identifier type.	This is because the trusted certificates in Sterling Integrator have issuer RDN information in the following format:
	`EmailAddress= sign@sc.com , Country=IN, Organization=Sterling Commerce, CommonName=UseThisForSigning, Country=IN, Organization=Sterling Commerce, =”`
	Most security modules expect it in the following format:
	`EmailAddress= sign@sc.com , Country=IN, Organization=Sterling Commerce, CommonName=UseThisForSigning”.`
	Sterling Integrator uses the same representation in Web services. Most crypto providers compare these using string comparison (for example, Merlin in case of Axis and the SecurityEnvironmentHandler defined for JavaEE). The user could use a crypto provider that can compare these by value (that is, the value of the individual name-value pairs in the issuer RDN).

Problem using attachments with Sterling Integrator Web services as a consumer on .NET.	.NET1.0 supports only DIME attachments and .NET2.0 supports MTOM, whereas Sterling Integrator only supports SwA (SOAP with Attachments, that is, MIME attachments). Currently Sterling Integrator cannot send or receive MIME attachments to or from .NET. However, it works well for inline attachments.

Axis2 does not generate stubs that take attachment as a parameter.	Axis2 supports attachments in a different way. Using WSTestUtil.java (below), change the stubs generated and add two methods (addAttachments and getAttachments) and modify those to suit your needs.
	`package com.sc.gis.testers;`
	`import java.io.FileOutputStream;`
	`import java.io.IOException;`
	`import java.io.StringWriter;`
	`import java.util.Properties;`
	`import javax.activation.DataHandler;`
	`import javax.activation.FileDataSource;`
	`import javax.xml.transform.OutputKeys;`
	`import javax.xml.transform.Transformer;`
	`import javax.xml.transform.TransformerFactory;`
	`import javax.xml.transform.dom.DOMSource;`
	`import javax.xml.transform.stream.StreamResult;`
	`import java.io.ByteArrayOutputStream;`
	`import java.io.File;`
	`import java.io.FileInputStream;`
	`import java.io.FileOutputStream;`
	`import java.io.InputStream;`
	`import java.net.URL;`
	`import java.util.Iterator;`
	`import javax.activation.DataHandler;`
	`import javax.activation.FileDataSource;`
	`import javax.xml.soap.AttachmentPart;`
	`import javax.xml.soap.MessageFactory;`
	`import javax.xml.soap.Name;`
	`import javax.xml.soap.SOAPBody;`
	`import javax.xml.soap.SOAPBodyElement;`

Question	Answer

	import javax.xml.soap.SOAPConnection;
	import javax.xml.soap.SOAPConnectionFactory;
	import javax.xml.soap.SOAPElement;
	import javax.xml.soap.SOAPEnvelope;
	import javax.xml.soap.SOAPHeader;
	import javax.xml.soap.SOAPMessage;
	import javax.xml.soap.SOAPPart;
	import javax.xml.ws.BindingProvider;
	import javax.xml.ws.Holder;
	import org.w3c.dom.Node;
	import com.sun.xml.xwss.SecurityConfigurationFactory;
	import com.sun.xml.xwss.XWSSecurityConfiguration;
	import org.apache.axiom.attachments.Attachments;
	import org.apache.axis2.context.MessageContext;
	import org.apache.ws.security.WSConstants;
	import org.apache.ws.security.handler.WSHandlerConstants;
	import org.w3c.dom.Document;
	public class WSTestUtil {
	 public static void printXML(Document doc) {
	 // set up a transformer
	 try {
	 TransformerFactory transfac =
	TransformerFactory.newInstance();
	 Transformer trans = transfac.newTransformer();
	 trans.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION,
	 "yes");
	 trans.setOutputProperty(OutputKeys.INDENT, "yes");
	 // create string from xml tree
	 StringWriter sw = new StringWriter();
	 StreamResult result = new StreamResult(sw);
	 DOMSource source = new DOMSource(doc);
	 trans.transform(source, result);
	 String xmlString = sw.toString();
	 System.out.println(xmlString);
	 } catch (Exception ex) {
	 ex.printStackTrace();
	 }
	 }
	
	 //copy paste
	 //WSTestUtil.addAttachments(_messageContext)
	 //WSTestUtil.getAttachments(_returnMessageContext);
	
	 //add this line to the stub's executeXXX method
	 //search for org.apache.axis2.context.MessageContext
	_messageContext
	 = new org.apache.axis2.context.MessageContext();
	 public static void addAttachments(MessageContext
	messageContext) {
	 FileDataSource dataSource = new
	FileDataSource("attachments\\outbound\\to_be_sent_axis2.txt");

Question	Answer

	`DataHandler dataHandler = new DataHandler(dataSource);`
	`messageContext.addAttachment("contentID$$$1234", dataHandler);`
	`}`
	`//add this line to the stub's executeXXX method`
	`//search for org.apache.axiom.soap.SOAPEnvelope _returnEnv = _returnMessageContext.getEnvelope();`
	`public static void getAttachments(MessageContext messageContext) {`
	`Attachments aMap = messageContext.getAttachmentMap();`
	`String soapContentId = aMap.getSOAPPartContentID();`
	`System.out.println("SOAP message Content Id:" + soapContentId);`
	`String[] idArr = aMap.getAllContentIDs();`
	`for (int i = 0; i < idArr.length; i++) {`
	`if ((idArr[i] != null) && !idArr[i].equals(soapContentId)) {`
	`DataHandler dh = aMap.getDataHandler(idArr[i]);`
	`try {`
	`FileOutputStream fos = new FileOutputStream(`
	`"attachments\\inbound\\attachmentAxis2$" + System.currentTimeMillis()+".txt");`
	`dh.writeTo(fos);`
	`fos.close();`
	`} catch (IOException ioe) {`
	`ioe.printStackTrace();`
	`}`
	`}`
	`}`
	`}`
	`public static void configureHTTPS()`
	`{`
	`System.setProperty("javax.net.ssl.trustStore", "certs\\httpsKeystore");`
	`System.setProperty("javax.net.ssl.trustStorePassword", "password");`
	`}`
	`public static boolean sendAndGetAttachmentLWJDBC(String endpointURL,String secureConfigFile)throws Exception`
	`{`
	`SOAPConnection connection = null;`
	`try`
	`{`
	`MessageFactory factory = MessageFactory.newInstance();`
	`//MessageFactory factory =MessageFactory.newInstance(SOAPConstants.SOAP_1_2_PROTOCOL);`
	
	`//MessageFactory factory =MessageFactory.newInstance(SOAPConstants.DYNAMIC_SOAP_PROTOCOL);`

Question	Answer

	`SOAPMessage message = factory.createMessage();`
	`SOAPPart soapPart = message.getSOAPPart();`
	`SOAPEnvelope envelope = soapPart.getEnvelope();`
	`SOAPHeader header = envelope.getHeader();`
	`header.detachNode();`
	`SOAPBody body = envelope.getBody();`
	`Name bodyName = envelope.createName("LightweightJDBCAdapterQuery", "mesa", "http://www.sterlingcommerce.com/mesa");`
	`SOAPBodyElement bodyElement = body.addBodyElement(bodyName);`
	`SOAPElement child = null;`
	`Name name = null;`
	`name = envelope.createName("sql","mesa","http://www.sterlingcommerce.com/mesa");`
	
	`child = bodyElement.addChildElement(name);`
	`child.addTextNode("SELECT * FROM MBX_MAILBOX");`
	`name = envelope.createName("pool","mesa","http://www.sterlingcommerce.com/mesa");`
	
	`child = bodyElement.addChildElement(name);`
	`child.addTextNode("mysqlPool");`
	`name = envelope.createName("result_name","mesa","http://www.sterlingcommerce.com/mesa");`
	
	`child = bodyElement.addChildElement(name);`
	`child.addTextNode("result");`
	`name = envelope.createName("row_name","mesa","http://www.sterlingcommerce.com/mesa");`
	
	`child = bodyElement.addChildElement(name);`
	`child.addTextNode("row");`
	`name = envelope.createName("query_type","mesa","http://www.sterlingcommerce.com/mesa");`
	
	`child = bodyElement.addChildElement(name);`
	`child.addTextNode("SELECT");`
	`//message.writeTo(System.out);`
	`AttachmentPart attachment = message.createAttachmentPart();`
	
	`FileDataSource fds = new FileDataSource(new File("attachments\\outbound\\to_be_sent_JavaEE.txt"));`
	`DataHandler dh = new DataHandler(fds);`
	`attachment.setDataHandler(dh);//Content("This is a sample attachment", "application/octetstream");`
	`attachment.setContentId("id111");`
	`message.addAttachmentPart(attachment);`
	`SOAPConnectionFactory soapConnectionFactory = SOAPConnectionFactory.newInstance();`

Question	Answer

	connection = soapConnectionFactory.createConnection();
	URL endpoint = new URL(endpointURL);
	if(secureConfigFile!=null)
	{
	FileInputStream f = new FileInputStream(secureConfigFile);
	
	XWSSecurityConfiguration config
	=SecurityConfigurationFactory.newXWSSecurityConfiguration(f);
	
	message.setProperty(XWSSecurityConfiguration.MESSAGE_SECURITY_CONFIGURATION,config);
	
	}
	SOAPMessage response = connection.call(message, endpoint);
	
	Iterator it = response.getAttachments();
	while(it.hasNext())
	{
	AttachmentPart inboundattachment =
	(AttachmentPart)it.next();
	DataHandler indh = inboundattachment.getDataHandler();
	FileOutputStream fos = new FileOutputStream(
	"attachments\\inbound\\attachmentJavaEE2$" +
	System.currentTimeMillis()+".txt");
	indh.writeTo(fos);
	fos.close();
	}
	response.writeTo(System.out);
	}
	catch(Exception ex)
	{
	ex.printStackTrace();
	return false;
	}
	finally
	{
	if(connection!=null)
	connection.close();
	}
	return true;
	}
	}

Why does Java EE give an exception while sending attachments?	A bug in Java EE causes it to give an exception while trying to send attachments. You can use the sample method in WSTestUtil.java (sendAndGetAttachmentLWJDBC) for sending attachments.

Copyright

Index