Informatica

Informatica® PowerExchange for Web Services

10.2

User Guide for PowerCenter

Informatica PowerExchange for Web Services User Guide for PowerCenter
10.2
September 2017

Publication Date: 2018-09-28

Table of Contents

Preface

The *Informatica PowerExchange® for Web Services User Guide for PowerCenter®* provides the information to extract data from a web service source, transform data using a web service, and load data into a web service target. It is written for database administrators and developers who are responsible for extracting data from web service sources, transforming data using web services, and loading data into web service targets.

This book assumes you have knowledge of web services concepts, including XML, Web Services Description Language (WSDL), and Simple Object Access Protocol (SOAP), relational database concepts, and PowerCenter.

Informatica Resources

Informatica Network

Informatica Network hosts Informatica Global Customer Support, the Informatica Knowledge Base, and other product resources. To access Informatica Network, visit https://network.informatica.com.

As a member, you can:

- Access all of your Informatica resources in one place.
- Search the Knowledge Base for product resources, including documentation, FAQs, and best practices.
- View product availability information.
- Review your support cases.
- Find your local Informatica User Group Network and collaborate with your peers.

Informatica Knowledge Base

Use the Informatica Knowledge Base to search Informatica Network for product resources such as documentation, how-to articles, best practices, and PAMs.

To access the Knowledge Base, visit https://kb.informatica.com. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Documentation

To get the latest documentation for your product, browse the Informatica Knowledge Base at https://kb.informatica.com/_layouts/ProductDocumentation/Page/ProductDocumentSearch.aspx.

If you have questions, comments, or ideas about this documentation, contact the Informatica Documentation team through email at infa_documentation@informatica.com.

Informatica Product Availability Matrixes

Product Availability Matrixes (PAMs) indicate the versions of operating systems, databases, and other types of data sources and targets that a product release supports. If you are an Informatica Network member, you can access PAMs at https://network.informatica.com/community/informatica-network/product-availability-matrices.

Informatica Velocity

Informatica Velocity is a collection of tips and best practices developed by Informatica Professional Services. Developed from the real-world experience of hundreds of data management projects, Informatica Velocity represents the collective knowledge of our consultants who have worked with organizations from around the world to plan, develop, deploy, and maintain successful data management solutions.

If you are an Informatica Network member, you can access Informatica Velocity resources at http://velocity.informatica.com.

If you have questions, comments, or ideas about Informatica Velocity, contact Informatica Professional Services at ips@informatica.com.

Informatica Marketplace

The Informatica Marketplace is a forum where you can find solutions that augment, extend, or enhance your Informatica implementations. By leveraging any of the hundreds of solutions from Informatica developers and partners, you can improve your productivity and speed up time to implementation on your projects. You can access Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

You can contact a Global Support Center by telephone or through Online Support on Informatica Network.

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
http://www.informatica.com/us/services-and-training/support-services/global-support-centers.

If you are an Informatica Network member, you can use Online Support at http://network.informatica.com.

CHAPTER 1

Understanding PowerExchange for Web Services

This chapter includes the following topics:

Understanding PowerExchange for Web Services Overview

PowerExchange for Web Services is a web service consumer that exchanges data with a web service provider. It integrates with PowerCenter to read data from a web service source and write data to a web service target. It also lets you use a web service to transform data during a session.

For example, you want to use PowerCenter to read loan request data from a database and run credit checks on the data, passing only those loan requests that pass the credit check to a target. Although you might not have the capability in-house to perform credit checks, you can process the loan request data in the mapping using an external web service. When you locate a web service that performs credit check operations, you can import it and use it as a Web Services Consumer transformation in a PowerCenter mapping. The PowerCenter Integration Service can connect to the web service during a session, so the web service can perform credit checks on the loan request data before the PowerCenter Integration Service writes the data to a target.

A web service is a collection of web service operations that you can access over an intranet or the Internet. Web service operations are programs that return data. When you access a web service, you request that the web service perform an operation and return data. A web service can contain many web service operations. For example, a web service that provides stock quotes might have a web service operation that returns the highest stock price of the day, a web service operation that returns the lowest stock price of the day, and a web service operation that returns the closing price for the day.

Web service operations contain input and output messages. Input and output messages are XML-formatted messages. They specify how to structure a request for a web service.

Web service access involves providers and consumers. A web service provider refers to the server that hosts the web service. A web service consumer refers to the client that requests a web service. PowerExchange for Web Services accesses web services as a web service consumer.

The web service you access can be remote or local. Someone at another organization can create and publish the web service, or someone at your organization can create and publish it.

Note: You only use PowerExchange for Web Services as a web service consumer. If you want to expose a PowerCenter workflow as a web service and make it available to others, you use PowerCenter Web Services Provider.

Before you can read data from a web service, write data to a web service, or transform data using a web service, you must import a web service operation. You import a web service operation from a Web Services Description Language (WSDL) file. WSDL files describe web services and web service operations. PowerExchange for Web Services uses the information in the WSDL file to access a web service operation.

PowerExchange for Web Services uses the Simple Object Access Protocol (SOAP) to exchange information with the web services provider and request web services. SOAP is a protocol for exchanging information between computers. It specifies how to encode XML data so that programs on different operating systems can pass information to each other. Web services hosts contain WSDL files and web services.

WSDL Files

WSDL files are XML documents that describe web services. A WSDL file contains the information that you need to access and use the web services that it describes. It describes web services that are supported by the web services host.

Note: The WSDL file can be located in a separate domain or on a local server. You indicate the WSDL file location by specifying a URL or a file path.

SOAP Encoding PowerExchange

WSDL files include information about how to encode SOAP request and response messages. SOAP encoding determines the format of the SOAP message body. Web service developers can use a variety of toolkits to create web services. Different toolkits support different ways of encoding SOAP messages.

PowerExchange for Web Services supports the following SOAP encoding styles:

- RPC/encoded
- Document/literal

You can import web service operations that support RPC/encoded or document/literal encoding styles. The WSDL use attribute is defined in the SOAP body. The use attribute value is either encoded or literal.

When the use is encoded, the WSDL must have a type attribute in the message part.

```
<message name="SystemNotAvailableException">
  <part type="ns3:SystemNotAvailableException" name="SystemNotAvailableException">
  </part>
```

When the use is literal, the WSDL must have an element attribute in the message part.

```
<message name="hypostoresPreferencesService_getAlertsProfile">
    <part name="getAlertsProfile" element="tns:getAlertsProfile">
    </part>
```

You can view the SOAP encoding style of the web service operation when you edit a web service source definition, a web service target definition, or a Web Services Consumer transformation.

WSDL File Component Hierarchy

WSDL files contain numerous components that describe interface, datatype, binding, and addressing information necessary for accessing and using a web service. PowerExchange for Web Services displays the following WSDL file components in the Import from WSDL (Web Services Consumer) window when you import a web service operation:

- **Service.** Contains groups of related ports. Each port defines an endpoint, which enables remote systems to connect to the service. The services section of the WSDL file defines each port's type, binding, and SOAP network address. Services contain one or more ports.
- **Port.** Defines the connection between bindings and the information required and returned by the web service. Each port defines a binding.
- **Binding.** Defines a protocol and data format for each operation. Each binding defines the format for one or more operations.
- **Operation.** Describes a program that performs an action, such as retrieving an order number. PowerExchange for Web Services supports request-response and one-way operations. Web service operations contain input and output messages. Input and output messages in the WSDL file contain the XML data that PowerExchange for Web Services uses to determine the groups and columns in the web service source and target definitions and Web Services Consumer transformations.

The following figure shows the relationship of WSDL file components:

A web service can contain numerous ports, which can contain numerous web service operations. The web service provider can make the same operation accessible from different ports. For example, the provider can bind an operation to a WSDL file with both a SOAP interface and an HTTP GET interface. This requires that two different ports contain the web service operation.

When you import a web service operation, you see the WSDL file component hierarchy in the Import from WSDL (Web Services Consumer) window so that you know which port contains the operation. You import a web service operation for a web service source definition, a web service target definition, and a Web Services Consumer transformation.

The legend in the Import from WSDL (Web Services Consumer) window indicates services, ports, bindings, and operations. You can only import web service operations. Services, ports, and bindings are not available to import.

Compression Support in SOAP Messages

The PowerCenter Integration Service can compress SOAP requests and responses to increase the speed and efficiency of sending and receiving messages over the network. To compress SOAP requests and responses, you need to implement compression and decompression at the PowerCenter Integration Service and at the web services host.

To implement compression of SOAP requests and response, the PowerCenter Integration Service uses the zlib compression technique in libcurl library. libcurl uses the compression techniques deflate and gzip to decode SOAP requests and responses. Also, libcurl requests all the supported compression techniques by setting a zero-length string in the Accept-Encoding header on the SOAP message. PowerExchange for Web Services uses gzip or deflate to process SOAP requests and responses.

Compression of SOAP Responses

Every SOAP message has an Accept-Encoding and a Content-Encoding header. The Accept-Encoding header field consists of a comma-separated list of encoding formats that PowerCenter Integration Service supports. These include gzip and deflate. Content-Encoding header specifies the compression algorithm that the web services host uses to compress SOAP responses.

To compress SOAP requests, the PowerCenter Integration Service adds a string in the Accept-Encoding header in the SOAP request. Based on the Accept-Encoding header that the PowerCenter Integration Service sets in the SOAP request, the web services host compresses SOAP responses and specifies the compression algorithm in the Content-Encoding header. libcurl reads the Content-Encoding header in the SOAP request and decodes the SOAP response using the corresponding compression algorithm.

Compression of SOAP Requests

You select the compression algorithm that the remote server interprets. You specify the compression algorithm in the session attribute SOAP Request Compression.

Integrating PowerCenter with Web Services

To integrate PowerCenter with web services, you can create web service source definitions, target definitions, and Web Services Consumer transformations. Web service source and target definitions provide the metadata for web service sources and targets. Web Service application connections enable the PowerCenter Integration Service to read data from web service sources and targets. Web Services Consumer transformations let the PowerCenter Integration Service use a web service to transform data.

You use the Designer to create web service source and target definitions and Web Services Consumer transformations. To create web service source and target definitions and Web Services Consumer transformations, the Designer imports metadata from a WSDL file. WSDL files describe web services and web service operations.

When you run a workflow to read or write web service data or transform data using a web service, the PowerCenter Integration Service connects to the server that hosts the web service defined in your mapping.

Designer and Web Services Integration

You use the Designer to import metadata that describes web service sources, targets, or transformations. It imports metadata by importing a web service operation from a WSDL file. The Designer can import a web service operation from an original WSDL file or from a copy of a WSDL file, located on a separate server from the original. It creates one source or target definition or transformation for each web service operation that you import.

Web service operations contain input and output messages. The Designer uses the input and output messages to determine the groups and columns in the web service source and target definitions and Web Services Consumer transformations.

Input and Output Messages

Input and output messages contain information about the structure of a web service. PowerCenter requires only output messages for a web service source definition. A web service source might require input values. Input messages are optional for web service sources. PowerCenter requires only input messages for a web

service target definition. It requires both input and output messages for a Web Services Consumer transformation. If the web service operation lacks the necessary input or output messages, the Designer does not let you import that operation.

The following table lists PowerCenter requirements for input and output messages:

Object	Message Requirement
Web service source definition	Output Input (optional)
Web service target definition	Input
Web Services Consumer transformation	Input and Output

PowerCenter Integration Service and Web Services Integration

When you run a session with PowerExchange for Web Services, the PowerCenter Integration Service communicates with the web service provider application or web server. This is the server that hosts the web service that you want to use in a PowerCenter session. The application or web server contains a web services host. The web services host contains a WSDL file and the web services that the WSDL file describes.

To communicate with a web services host, the PowerCenter Integration Service requires an endpoint URL. The PowerCenter Integration Service uses the endpoint URL defined in the Web Service application connection or contained in the WSDL file as the location attribute. You can configure a Web Service application connection in the Workflow Manager. You can also configure a web service source, target, or transformation to use a dynamic endpoint URL.

SOAP Messages

When you read data from a web service source, write data to a web service target, or transform data with a Web Services Consumer transformation, the PowerCenter Integration Service sends a SOAP request to a web services host. When the PowerCenter Integration Service sends a SOAP request, it requests a web service to perform a specified operation. The web service operation returns data to the PowerCenter Integration Service in a SOAP response.

Note: PowerExchange for Web Services supports only the SOAP protocol for requesting a web service.

The PowerCenter Integration Service sends SOAP requests to a web services host over HTTP. PowerExchange for Web Services supports only HTTP and HTTPS for SOAP request and response document transport.

When you use PowerExchange for Web Services to read, write, or transform web service data, the PowerCenter Integration Service uses the endpoint URL defined in the Web Service application connection or contained in the WSDL file to connect to the web services host. When the PowerCenter Integration Service connects to the web services host, it generates and sends a SOAP request using the metadata specified in the mapping. The SOAP request contains the information necessary to execute the web service. It contains any input data that the web service requires to perform the operation.

Reading Data from a Web Service Source

After the web service performs the operation, the web services host sends a SOAP response that contains the data that results from the operation. The PowerCenter Integration Service reads the SOAP response and passes the data through the pipeline.

Writing Data to a Web Service Target

After the PowerCenter Integration Service successfully connects to the web services host and executes the web service, it writes data to the web service target. The PowerCenter Integration Service ignores any SOAP response that the web services host sends.

Transforming Data Using a Web Services Consumer Transformation

After the web service performs the operation, the web services host sends a SOAP response that contains the data that results from the operation. The PowerCenter Integration Service receives the SOAP response and passes the data to a target.

Using Code Pages

WSDL files contain an XML encoding declaration that indicates the code page that the web service uses. The most commonly used code pages in XML are UTF-8 and UTF-16. All XML parsers support these two code pages. For information on the XML character encoding specification, go to the W3C website at http://www.w3c.org.

PowerCenter supports the same set of code pages for web services that it supports for relational databases and other flat files. Use any code page supported by both Informatica and the XML specification. Informatica does not support any user-defined code pages.

For web service source and target definitions and Web services Consumer transformations, PowerCenter uses the code page declared in the WSDL file. If Informatica does not support the declared code page, the Designer returns an error.

Security and Web Services

When a web service consumer or a web service provider sends or receives data over a network, the data is subject to security risks. Both web service consumers and providers share the following security concerns:

- **Authentication**. Web service providers and consumers must verify the identity of each user before transmitting data. They must also verify the origin of data before transmitting it.
- **Confidentiality**. Web service providers and consumers must prevent third parties from deciphering any intercepted data.
- **Data integrity.** Web service providers and consumers must ensure that data has not been lost, modified, or destroyed during transmission.

The following primary types of security are available to address these concerns:

- **Message-layer security**. Security embedded in a web service message. Message-layer security can include encryption to secure SOAP messages. It can also include certificates and security tokens for authentication and confidentiality.
- **Transport-layer security**. Security implemented on top of the transport layer (TCP layer) of TCP/IP using Secure Sockets Layer (SSL). Transport-layer security enables web services to use Hypertext Transfer Protocol over SSL (HTTPS) as a web address for secure message transport.

PowerExchange for Web Services provides message-layer security by adding a WSSE security header which contains authentication information for the web service provider to authenticate the PowerCenter Integration Service.

PowerExchange for Web Services provides transport-layer security in the following situations:

- When importing a WSDL file from a web server that uses transport-layer security.
- When the PowerCenter Integration Service sends a web service request to a web server and receives a response.

SSL Authentication Components

PowerExchange for Web Services ensures authentication by using the Public Key Infrastructure (PKI) standard, which includes the following components:

- **Authentication certificate**. A digital certificate that a certificate authority provides to verify and authenticate parties in Internet communications. A certificate authority is a trusted, independent third party that issues digital certificates. Digital certificates are attachments to electronic messages, such as SOAP messages, used for security. They use public keys to encrypt messages and send authentication information. Message recipients also use public keys to verify and authenticate the sender and decode and view messages. Recipients can use these keys to send encrypted replies.
- **Trust store**. A file that contains authentication certificates that the PowerCenter Integration Service uses to authenticate requests from web service providers. You can store multiple authentication certificates in the trust store. By default, the trust certificates file for PowerExchange for Web Services is named ca-bundle.crt. It contains certificates issued by major, trusted certificate authorities, such as VeriSign. You can add certificates to the ca-bundle.crt file.
- **Client store**. A file that contains authentication certificates that the PowerCenter Integration Service sends to web service providers for authentication. You can store multiple authentication certificates in the client store. The web service provider uses the public key contained in the certificate to send an encrypted reply to the PowerCenter Integration Service. The PowerCenter Integration Service then uses its private key to decrypt the reply and respond to the web service. This process, called an SSL handshake, enables the web service provider to authenticate communication with the PowerCenter Integration Service.

During a session with a web service provider that requires authentication, the PowerCenter Integration Service and the web service provider authenticate each other using authentication certificates before either can transmit data. When a web service provider sends an authentication certificate to the PowerCenter Integration Service, the PowerCenter Integration Service verifies that the authentication certificate exists in the trust store before it authenticates the web service provider. When the PowerCenter Integration Service authenticates to a web service provider, it sends an authentication certificate from its client store.

Types of SSL Authentication

The PowerCenter Integration Service is a web service client. During a web service session, the PowerCenter Integration Service connects to a web service provider. The web service provider uses the following types of SSL authentication to authenticate PowerCenter Integration Service:

- **Client authenticating server.** When the PowerCenter Integration Service connects to a web service provider during a web service session, it establishes an SSL session to authenticate the web service provider. The web service provider sends an authentication certificate to the PowerCenter Integration Service. The PowerCenter Integration Service verifies that the authentication certificate exists in the trust certificates file. For this type of authentication, you configure the trust certificates file.
- **Server authenticating client.** When the PowerCenter Integration Service connects to a web service provider during a web service session, the web service provider establishes an SSL session to authenticate the PowerCenter Integration Service. At the request of the web service provider, the

PowerCenter Integration Service sends a client certificate file containing a public key. The web service provider uses the public key for authentication and verifies that the PowerCenter Integration Service can be trusted. For this type of authentication, you configure the client certificate file and the corresponding private key file.

- **Mutual authentication.** When you establish an SSL session with mutual authentication, the PowerCenter Integration Service and the web service provider exchange certificates and verify that they can trust each other. For mutual authentication, you configure the trust certificates file, the client certificate, and the corresponding private key file.

Transport-Layer Security

Web service providers establish transport-layer security using HTTPS. HTTPS uses SSL to provide the following security functions:

- Enable web servers and web browsers to authenticate users before transmitting data.
- Enable web servers to communicate using a secure connection.
- Provide data encryption and prevent third parties from intercepting data during transmission.

When you import web service definitions and Web Services Consumer transformations from a WSDL file located on a web server and the URL for the web server begins with HTTPS, the web service provider uses SSL. Similarly, when an endpoint URL configured in a Web Service application connection or contained in a WSDL file begins with HTTPS, the web service provider uses SSL. During a session with web service definitions or Web Services Consumer transformations imported from a web service provider using SSL, the PowerCenter Integration Service reads and writes data using transport-layer security.

Note: When you import a WSDL file or send a web service request to a web service provider that requires authentication, you must provide a user name and password. You can configure a Web Service application connection for a web server with a user name and password.

Authentication in Web Services

When you import a web service definition or a Web Services Consumer transformation, the external web service provider authenticates the Designer. When you connect to an external web service provider to read or write data, the external web service provider authenticates the PowerCenter Integration Service.

The web service provider uses the following types of authentication:

- **Basic authentication.** Requires you to provide a user name and password for the domain of the web service provider. The PowerCenter Integration Service sends the user name and the password to the web service provider for authentication.
- **Digest authentication.** Requires you to provide a user name and password for the domain of the web service provider. The PowerCenter Integration Service generates an encrypted message digest from the user name and password and sends it to the web service provider. The web service host generates a temporary value for the user name and password and stores it in the Active Directory on the Domain Controller. It compares the value with the message digest. If they match, the web service provider authenticates you.
- **NTLM authentication.** Requires you to provide a domain name, server name, or default user name and password. The web service provider authenticates you based on the domain you are connected to. It gets the user name and password from the Windows Domain Controller and compares it with the user name and password that you provide. If they match, the web service provider authenticates you. NTLM authentication does not store encrypted passwords in the Active Directory on the Domain Controller. The web service provider uses NTLM v1 or NTLM v2 for authentication.

The PowerCenter Integration Service sends authentication requests to the web services host over HTTP. The web services host sends a response with the status code 401 with a header that indicates the types of authentication it supports. The PowerCenter Integration Service sends the login credentials to the web services host. To make HTTP calls, the PowerCenter Integration Service uses the curl library, which performs the authentication.

WSSE Security

The PowerCenter Integration Service can also include a WSSE security header when it sends a SOAP request to the web service provider. The WSSE security header contains authentication information so the web service provider can authenticate the PowerCenter Integration Service. WSSE security header also works with basic, digest, and NTLM authentication types.

When you import a Web Services Consumer transformation or a web service target definition, you can select a WSSE security type. The Designer adds WSSE security header information to the target definition or transformation based on the security type you select. To use a WSSE security header for a web service source, you need to manually add the header in the SOAP request.

The following sample code shows a WSSE security header in a SOAP 1.1 and SOAP 1.2 request:

```
<SENC:Envelope xmlns:SENC="..." xmlns:wsse="...">
<SENC:Header>
<wsse:Security>
<wsse:UsernameToken>
<wsse:Username>admin</wsse:Username>
<wsse:Password>admin123</wsse:Password>
</wsse:UsernameToken>
</wsse:Security>
</SENC:Header>
</SENC:Envelope>
```

Cookie Authentication

You can configure the Web Services Consumer transformation to use cookie authentication. With this type of authentication, the web service provider sends a packet of information called a cookie to the PowerCenter Integration Service. The PowerCenter Integration Service returns the cookie each time it accesses the web service provider. You configure cookie authentication by using the Cookie port of the Web Services Consumer transformation.

POODLE Vulnerability Fix

The SSL 3.0 POODLE vulnerability affects Informatica 9.6.x server-side and client-side components. The vulnerability affects secure communication within the Informatica domain between services and client applications.

In version 9.6.1 HotFix 2, Informatica took the following steps to address POODLE vulnerability:

- Disabled the SSL 3.0 protocol in Informatica libraries.
- Updated the OpenSSL libraries packaged with Informatica products to include support for TLS_FALLBACK_SCSV.

You can upgrade to Informatica 9.6.1 HotFix 1, or you can apply the following EBFs: EBF 14580, EBF 14579, and EBF 14578. You can download the EBFs from https://tsftp.informatica.com/ at the following locations:

- /Informatica9/9.6.1 HotFix1/
- /Informatica9/9.6.1/

Note: Before you download and apply an EBF, verify that the EBF is applicable to your setup. For information, contact Informatica Global Customer Support.

If you are installing or upgrading to Informatica, you must apply the following EBFs:

- EBF 14580 for Informatica 9.1.0 HotFix 6.
- EBF 14579 for Informatica 9.5.1 HotFix 4.
- EBF 14578 for Informatica 9.6.1 HotFix 1.

CHAPTER 2

Configuring PowerExchange for Web Services

This chapter includes the following topics:

Configuring PowerExchange for Web Services Overview

PowerExchange for Web Services requires installation and configuration on the PowerCenter Integration Service, PowerCenter Client, and PowerCenter Repository Service.

To configure PowerExchange for Web Services, complete the following steps:

1. **Configure HTTP proxy options (Optional).** Configure the HTTP proxy options for the PowerCenter Integration Service.
2. **Configure certificates (Optional).** Configure certificates for SSL authentication.

Before you configure PowerExchange for Web Services, install or upgrade PowerCenter.

Step 1. Configure HTTP Proxy Options for the PowerCenter Integration Service

You can optionally configure properties for the HTTP proxy server for the PowerCenter Integration Service. When you configure HTTP proxy options, set the following properties in the Informatica Administrator:

- HttpProxyServer
- HttpProxyPort
- HttpProxyUser

- HttpProxyPassword
- HttpProxyDomain

Step 2. Configure Certificates for SSL Authentication

Before you configure a Web Services application connection to use SSL authentication, you may need to configure certificate files. If the PowerCenter Integration Service authenticates the web service provider, you configure the trust certificates file. If the web service provider authenticates the PowerCenter Integration Service, you configure the client certificate file and the corresponding private key file, passwords, and file types. You can generate client certificate and private key files by running the OpenSSL commands.

The trust certificates file (ca-bundle.crt) contains certificate files from major, trusted certificate authorities. If the certificate bundle does not contain a certificate from a certificate authority that the web service provider uses, you can convert the certificate of the web service provider to PEM format and append it to the ca-bundle.crt file.

The private key for a client certificate must be in PEM format.

Generate Client Certificate and Private Key Files

If the web service provider authenticates the PowerCenter Integration Service, you must configure the client certificate and private key files. You can generate client certificate and private key files and use these files to configure the Web Service Consumer application connection.

You can generate the client certificate and private key files in a single file or as separate files.

Generate One Certificate File

To generate the client certificate file and private key file in a single file, use the following command:

```
openssl pkcs12 -in <certificate authority file>.p12 -out test1.pem -clcerts
```

The command generates a single certificate file in the PEM format. In the Web Service Consumer application connection, use the single certificate file while configuring both the client certificate file and the private key file. Use the password that you provide after running the OpenSSL command to configure the Web Service Consumer application connection.

Generate Keys in Separate Files

- To generate the client certificate file, use the following command:

  ```
  openssl pkcs12 -in <certificate authority file>.p12 -nokeys -out clientcert.pem
  ```

- To generate the private key file, use the following command:

  ```
  openssl pkcs12 -in <certificate authority file>.p12 -nocerts -out pk.pem
  ```

The command generates certificate files in the PEM format. In the Web Service Consumer application connection, specify the fully qualified path along with the client certificate and private key files. Use the passwords that you provide after running the OpenSSL commands to configure the Web Service Consumer application connection.

Configure the Web Service Consumer Application Connection

Use the client certificate file, the corresponding private key file, and the passwords to configure a Web Service application connection to use SSL authentication. You can access the Web Service application connection from the Application type connection in the Workflow Manager.

Converting Certificate Files from Other Formats

Certificate files have the following formats:

- **DER.** Files with the .cer or .der extension.
- **PEM.** Files with the .pem extension.
- **PKCS12.** Files with the .pfx or .P12 extension.

When you append certificates to the ca-bundle.crt file, the certificate files must use the PEM format. Use the OpenSSL utility to convert certificates from one format to another. You can obtain OpenSSL at http://www.openssl.org.

For example, to convert the DER file named server.der to PEM format, use the following command:

```
openssl x509 -in server.der -inform DER -out server.pem -outform PEM
```

If you want to convert the PKCS12 file named server.pfx to PEM format, use the following command:

```
openssl pkcs12 -in server.pfx -out server.pem
```

To convert a private key named key.der from DER to PEM format, use the following command:

```
openssl rsa -in key.der -inform DER -outform PEM -out keyout.pem
```

After you convert certificate files to the PEM format, you can append them to the trust certificates file. Also, you can use PEM format private key files with PowerExchange for Web Services.

Adding Certificates to the Trust Certificates File

If the web service provider uses a certificate that is not included in the ca-bundle.crt file, you can add the certificate to the ca-bundle.crt file.

To add a certificate to the trust certificates file:

1. Use Internet Explorer to locate the certificate and create a copy:
   - Access the web service provider using HTTPS.
   - Double-click the padlock icon in the status bar of Internet Explorer.
   - In the Certificate dialog box, click the Details tab.
   - Select the Authority Information Access field.
   - Click Copy to File.
   - Use the Certificate Export Wizard to copy the certificate in DER format.
2. Convert the certificate from DER to PEM format.
3. Append the PEM certificate file to the certificate bundle, ca-bundle.crt.

   For more information about adding certificates to the ca-bundle.crt file, see the curl documentation at http://curl.haxx.se/docs/sslcerts.html.

Registering the Plug-in

If you upgrade PowerExchange for Web Services for PowerCenter from a previous version to the latest version, you must update the plug-in registration when you register the plug-in.

A plug-in is an XML file that defines the functionality of PowerExchange for Web Services. To register the plug-in, the PowerCenter Repository Service must run in the exclusive mode. Use the Informatica Administrator or `pmrep` RegisterPlugin command to register the plug-in.

Note: If you do not have the privileges to register the plug-in, contact the user who manages the PowerCenter Repository Service.

CHAPTER 3

Web Service Sources and Targets

This chapter includes the following topics:

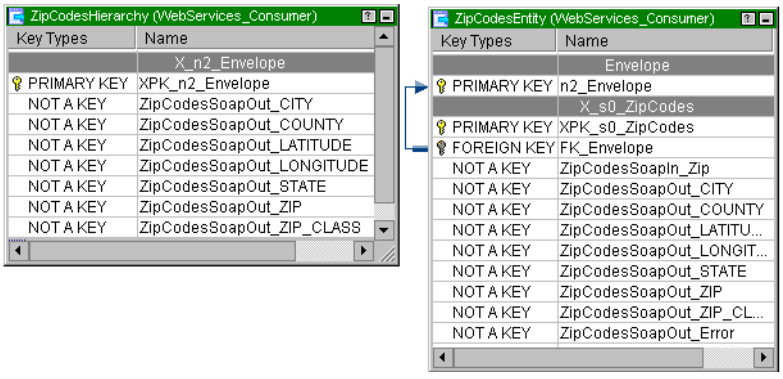
Web Service Sources and Targets Overview

Web service source and target definitions represent metadata for web service operations. A web service operation contains input or output messages in XML format. Input and output messages describe the data that web service operations exchange with the service that is running.

When you import a web service source or target definition, you import a web service operation from a Web Services Description Language (WSDL) file.

Web service source and target definitions contain one or more groups and one or more columns. The structure of the source or target definition depends on the input or output messages for the operation imported from the WSDL file. The Designer creates primary and foreign keys to indicate the relationships between groups.

The following figure shows groups, column names, and keys in a web service source definition with multiple groups. It shows the same source definition with one group:



Step 2 of the wizard determines whether the source has one group or multiple groups. The default choice is Entity, and this results in multiple groups in the source. Hierarchy-normalized results in one group. - After you choose one of them, click Finish.

The Designer gives a web service source and target definition the same name as the web service operation that you import. When you import a web service source or target definition, the Designer places it under the Sources or Targets node. The Designer places the web service source definition in a WebServices_Consumer database definition node.

Use any of the following methods to create a web service definition:

- Import a web service operation from a remote WSDL file located on a URL.
- Import a web service operation from a local WSDL file.

After you create the web service source or target, you can edit the definition in the Designer workspace or the WSDL workspace. When you import the web service source or target definition from a WSDL, you can view the web service source or target definition and edit a limited number of properties in the Designer workspace. You can edit the web service source or target definition in the WSDL workspace. If you create an empty source or target definition, you can define the views and ports in the WSDL workspace.

Supported Web Service Operations

Any web service operation that you import for a web service source or target definition must have the proper encoding, web service operation type, and input or output messages. You can import all web service operations with the following characteristics:

- Specifies the request-response type for web service source definitions or web service target definitions. PowerExchange for Web Services also supports the one-way type for web service targets only.
- Contains either RPC/encoded or document/literal SOAP encoding style.
- WSDL file specifies either the HTTP or HTTPS transport protocol.

If the web service operation that you want to import differs from this list of supported characteristics, you cannot import it.

WSSE Security for Web Service Targets

You can select a WSSE security type when you create a web service target definition.

The following table shows the available WSSE security types:

WSSE Security Type	Description
None	The PowerCenter Integration Service does not add the WSSE security header to the generated SOAP request. Default is none.
PasswordText	The PowerCenter Integration Service adds security header for WSSE security authentication to the generated SOAP request. Password is stored in the clear text format. You can also provide a base64 encoded hash password.
PasswordDigest	The PowerCenter Integration Service adds security header for WSSE security authentication. Password is stored in a digest form which provides effective protection against replay attacks over the network. The PowerCenter Integration Service also adds NONCE (a random generated value that is valid only once for that specific username token) and CREATED (timestamp of the Username token in UTC timezone format) to the generated SOAP request.

If you select a WSSE security type, the Designer adds the fields wsse_Username and wsse_Password to the web service target definition. You can pass values for the user name and password from upstream transformations.

When the PowerCenter Integration Service creates a SOAP request to connect to the web service, the PowerCenter Integration Service adds a WSSE security header. The WSSE security header contains the authentication information in the wsse_Username and wsse_Password fields. The format of the user name and password is based on the security type you selected.

Cookie Columns and URL Columns for Source and Target Definitions

When you import a WSDL file and create a web service source or target definition, you can create one or both of the following columns:

- **Cookie column**. Creates a cookie column that can accept cookies and passes them to subsequent PowerCenter Integration Service calls. The PowerCenter Integration Service uses the cookie to authenticate subsequent calls in other web service sources and targets. Create a cookie column when a remote web server implements user sessions based on cookies.
- **URL column**. Creates a URL column that can receive information about authenticated URLs and pass them to subsequent PowerCenter Integration Service calls. The PowerCenter Integration Service uses the information to authenticate subsequent calls in other web services sources and targets. Create a URL column when you want to pass a dynamically generated endpoint URL instead of a static endpoint URL to PowerExchange for Web Services. The value in this column overrides the endpoint URL that you specify in the source definition or target definition properties.

Rules and Guidelines for Creating Web Service Definitions

Use the following rules and guidelines when you import or create web service source and target definitions:

- **Use a WSDL for elements with complex relationships.** To create a web service source or target definitions with a complex element relationship, first create a WSDL to define the element hierarchy and then import the source or target definition from the WSDL. Use a WSDL to create a web service source or target definition that contain multiple occurrences of elements or that contain elements of complex type.
- **Use a WSDL with a global element.** When you import a definition from a WSDL that has no global element, the Designer cannot create a root view in the web service definition. The Designer displays a message that there is no global element.
- **Use a WSDL to create targets with fault views.** If you want the target definition to have fault views for specific data error, use a WSDL to create the web service target definition.
- **The input and output message in the WSDL must have the same encoding style.** If you import web service source and target definition from a WSDL, the encoding style for the input and output messages must be the same. If the input message uses the RPC/SOAP Encoded style, the output message must also use the RPC/SOAP Encoded style. If the input message uses the Document/Literal style, the output message must also use the Document/Literal style.

XML Views and Groups

Web service source and target definitions are organized into XML views. XML views are groups of columns that represent the elements and attributes that define the input and output messages.

When you import web service source and target definitions from a WSDL or create the source and target definitions from columns, the Designer generates the views based on the type of relationship between the elements in the input or output messages and on the definition of those elements.

The web service source and target definitions can contain the following views:

- **Envelope.** Main view that contains a primary key and the ports for the input or output message. For a simple WSDL or a simple list of columns, the Designer typically generates only an envelope view.

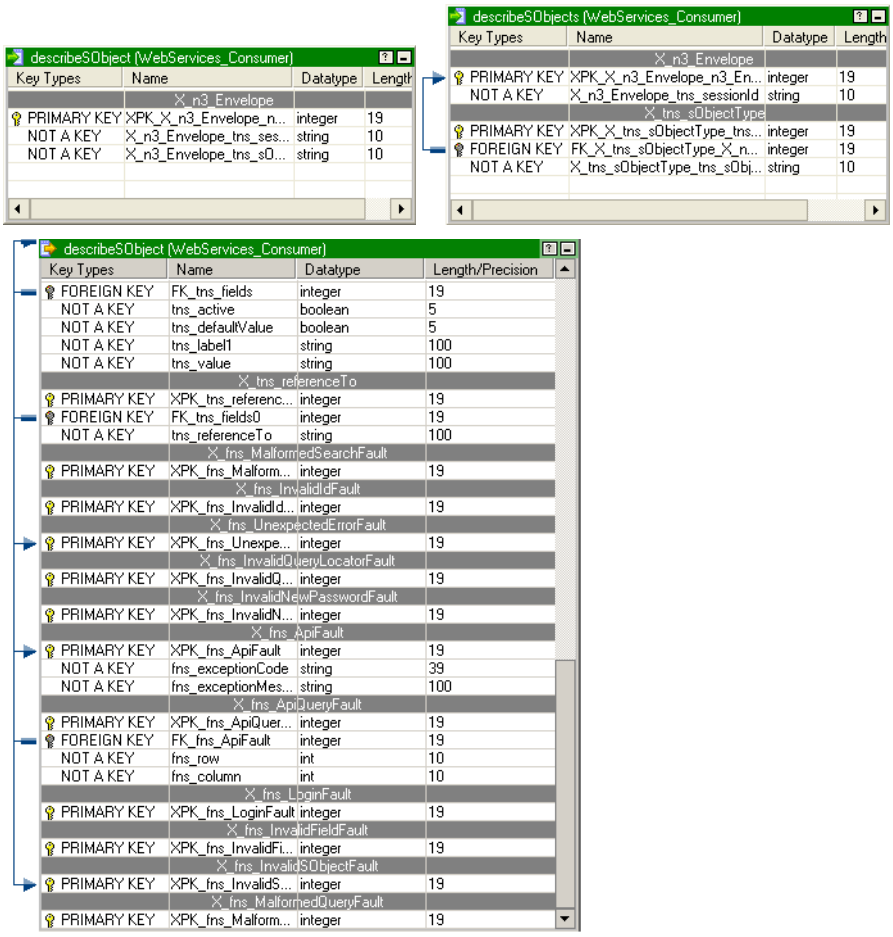
  The Designer generates an envelope view for web service source and target definitions with elements that have a normalized hierarchical relationship or entity relationship.
- **Element.** View created if the input or output message contains a multiple occurring element. The Designer generates an element view for each multiple occurring element in the input or output message. The element view has an n:1 relationship with the envelope view.

  The Designer generates an element view for web service source and target definitions with elements that have a normalized hierarchical relationship or entity relationship.
- **Type.** View created if the input or output message contains a definition of a complex type. The Designer generates a type view for each complex type element in the input or output message. The type view has an n:1 relationship with the envelope view.

  The Designer generates a type view for web service source and target definitions with elements that have an entity relationship.
- **Fault.** View created if a fault message is defined for the output message of the operation. The Designer generates a fault view for each fault message defined for the operation. The fault view has an n:1 relationship with the envelope view. Only web service target definitions contain fault views.

  The Designer generates a fault view for web service target definitions with elements that have a normalized hierarchical relationship or entity relationship.

The following source and target definitions show examples of the XML views generated for web service source and target definitions:



Source Definition

The Designer generates XML views for the web service source definition based on the definition of the input message associated with the operation.

Normalized Hierarchical Relationship

The Designer generates the following views for a source definition with a normalized hierarchical relationship:

- Envelope
- Element

Entity Relationship

The Designer generates the following views for a source definition with an entity relationship:

- Envelope
- Element
- Type

Target Definition

The Designer generates XML views for the web service target definition based on the definition of the output message or any fault message associated with the operation. Because a function within an operation can result in different faults, the Designer may create multiple fault views in the target definition. A fault message represents an error processing the request.

Normalized Hierarchical Relationship

The Designer generates the following views for a target definition with a normalized hierarchical relationship:

- Envelope
- Element
- Fault

Entity Relationship

The Designer generates the following views for a target definition with an entity relationship:

- Envelope
- Element
- Type
- Fault

Element Relationships

When you import a web service source or target definition from a WSDL, you can create XML views with the following types of element relationships:

- **Normalized hierarchical relationship.** This is the default option for source or target definitions imported from a WSDL file. In a normalized hierarchical view, every element or attribute appears once. One-to-many relationships become separate XML views with keys to relate the views.
- **Entity relationship.** Use this option to create relationships between views instead of one large hierarchy. When you create a web service source or target with an entity relationship, the Designer generates separate views for multiple-occurring elements and complex types. The Designer includes views for all derived complex types.

Importing a Web Service Source or Target Definition

Follow the same steps to import a web service source or target definition from a WSDL. Since the source and target definitions represent different elements in the WSDL, the source definition created by the Designer differs from the target definition.

You can import a web service source or target from a WSDL that you can access locally or through a URL. You can import definitions from a WSDL file with RPC/Encoded or Document/Literal styles. You can also import definitions from a WSDL file over an HTTPS connection. The Designer can import web service source or target definitions from nested WSDL file definitions where the WSDL file contains information to import other WSDL files.

To import a web service source or target definition:

1. To import a web service source definition, in the Source Analyzer, choose Sources > Import from WSDL (Consumer). Or, to import a web service target definition, in the Target Designer, choose Targets > Import from WSDL (Consumer).

   The Import from WSDL (Web Services Consumer) dialog box appears.

2. Click Advanced Options to configure the default precision for String datatype fields and to set column naming conventions.

   The Change XMLViews Creation and Naming Options dialog box appears.

   Configure the following options:

Option	Description
Override all infinite lengths	You can specify a default length for fields with undefined lengths, such as strings. By default, this option is selected.

Generate names for XML columns	You can choose to name XML columns with a sequence of numbers or with the element or attribute name from the schema. If you use names, choose from the following options:
	- When the XML column refers to an attribute, prefix it with the element name. PowerCenter uses the following format for the name of the XML column:
	`NameOfElement_NameOfAttribute`
	- Prefix the XML view name for every XML column. PowerCenter uses the following format for the name of the XML column:
	`NameOfView_NameOfElement`
	- Prefix the XML view name for every foreign-key column. PowerCenter uses the following format for the name of a generated foreign key column:
	`FK_NameOfView_NameOfParentView_NameOfPKColumn`
	Maximum length for a column name is 80 characters. PowerCenter truncates column names longer than 80 characters. If a column name is not unique, PowerCenter adds a numeric suffix to keep the name unique.

Default length for anyType element mapped to string	Default length of the string port created for an element of type anyType. You can create a port of type string for an element of type anyType. By default, the length of the string is the value you set here. To change the length of the string, edit the web service source or target definition in the WSDL workspace. Default is 10,000.

   After you configure these options, the Designer automatically applies them to all web services source and target definitions you create.

3. Click URL to import from a remote WSDL file located on a URL. Or, choose to import from a local file or a URL.

   If you import from a local file, select a WSDL file in a local folder and click Open.

   If you import from a URL, type a URL or select a URL from the Address list and click Open.

   Or, click UDDI to import from a remote WSDL file listed in a UDDI directory. Reserved for future use.

4. If you are importing from a remote WSDL file located on a URL, enter the URL in the Address field. Or, if you are importing from a local WSDL file, navigate to the directory that contains the WSDL file and select the WSDL file.

5. Click Open.

If you did not configure the advanced settings in step 2, the Designer asks if you want to override the infinite length option. Click Yes to open the Change XMLViews Creation and Naming Options dialog box to configure advanced options.

6. Select the operation defined in the WSDL for which you want to create the source or target definition.

   Step 1 of the Web Services Wizard appears.

7. Click Next.

   Step 2 of the Web Services Wizard appears.

8. Choose whether you want to generate the XML views as entity relationships or as normalized hierarchical relationships.

   If you select Hierarchy Relationships, then Normalized XML Views is automatically selected. The Denormalized XML Views option is reserved for future use.

   If the Designer determines that it will generate more than 400 views from a WSDL schema, the Designer does not generate the definition. You can import the WSDL schema and manually create the XML views in the WSDL workspace.

   When you import a definition from a WSDL schema that has no global elements, the Designer cannot create a root view in the definition. The Designer displays a message that there is no global element.

   Optionally, select Create Cookie Port to create a cookie column for the source or target definition.

9. Optionally, select Create URL Port to create a URL column for the source or target definition.
10. Click Finish.

Creating a Web Service Source or Target Definition

When you import a source or target definition from a WSDL but select the option not to create XML views, the Designer creates an empty definition. After the definition is created, right-click the title of the source or target definition and select the last item labeled WSDL Workspace to define the views and ports and the relationships between views.

Importing from a WSDL Without Creating XML Views

You might want to create an empty source or target definition if you have a WSDL that defines a large number of elements and you do not want to include all elements in the definition. For example, you might have a WSDL that defines ten elements in the input message but you want to include only two of the elements in your source definition. You can create an empty source definition and manually define the two elements.

To import a source or target definition from a WSDL without creating XML views, select the Do not create XML views option in step 2 of the import process.

After you create an empty source or target definition, use the WSDL workspace to define XML views and columns, and the relationship between views.

Editing a Web Service Source or Target Definition

After you import a web service source or target definition, you can edit the definition to change the SOAP action, SOAP request for web service source definitions, column precision values, and other properties. The Designer propagates the changes to any mapping that uses an instance of the source or target definition.

To edit a web service source or target definition:

1. To edit a web service source definition, in the Source Analyzer, double-click the title bar of the source definition. Or, to edit a web service target definition, in the Target Designer, double-click the title bar of the target definition.

   The Edit Tables dialog box appears.
2. Click the Columns tab.
3. Click the Web Services Consumer Properties tab.
4. Optionally, edit the Web Services Consumer Properties settings:

Dialog Settings	Description
Operation Type	Web service operation encoding type. PowerExchange for Web Services supports RPC/encoded and document/literal encoding types.
Original WSDL Location	URL for the WSDL file from which you originally imported the web service source or target definition.
Operation Name	Name of the web service operation for the web service source or target definition.
SOAP Action	Valid SOAP action defined in the WSDL file. This is required for HTTP binding of a web service.
End Point URL	Endpoint URL for the web service host that you want to access. You can use a mapping parameter or variable as the endpoint URL. For example, you can use mapping parameter "$$MyURL" as the endpoint URL, and set $$MyURL to the URL in the parameter file.
SOAP Request	Displays the SOAP request that the PowerCenter Integration Service sends to a web services host to read from a web service source. This field is visible for web service source definitions only. You can edit the SOAP request if the web service source requires input values. Manually edit the SOAP request if you have experience editing XML documents. Otherwise, click Populate SOAP Request for assistance with generating the request. When you manually edit the SOAP request, you can click Apply to apply the changes. **Note:** If you edit the SOAP request, keep a backup of the original SOAP request.

5. Optionally, click Populate SOAP Request to modify the SOAP request for the Web Service source definition.

   If you are editing a web service target definition, go to step 9.

   The Populate SOAP Request dialog box appears.
6. Select an element or attribute to change its value.

   The element you select appears in the Tree Item Value field. Elements and attributes that take a value use the following syntax when they display in the Populate SOAP Request dialog box:

   ```
   [datatype] Element/AttributeName = [value]
   ```
7. Edit the Tree Item Value.

For array type nodes you can add, update, and delete elements from an array.

8. Click OK.

   The Designer generates a valid SOAP request that contains the input values.

9. Optionally, select the WS Security Type for the web service target definition.
10. Click OK.

Viewing the SOAP Version

After you import a web service source or target definition, you can view the SOAP version of the web service operation.

1. In the Source Analyzer or Target Designer, double-click the title bar of a web service source or target definition.

   The **Edit Tables** dialog box appears.

2. Click the **Metadata Extensions** tab.

You can view the SOAP version of the web service operation.

Viewing WSDL Group Details

After you import a web service source or target definition or a Web Services Consumer transformation, you can view the details of each group in the WSDL Workspace. The WSDL Workspace shows datatype and hierarchy information for group components, the relationships of the components of a complex WSDL schema, and details about the following XML group components:

- Simple types
- Complex types
- Attributes
- Attribute groups
- Elements
- Element groups
- Lists
- Unions
- Local Declarations

Most components in the WSDL Workspace are read-only. However, you can create and delete reference ports that refer to rough ports that you added to Web Services Consumer transformations.

To view the WSDL Workspace:

1. Right-click the title bar of a web service source definition, target definition, or Web Services Consumer transformation.
2. Choose WSDL Workspace.

   The WSDL Workspace appears.

The WSDL Workspace uses the XML Editor.

Editing Definitions in the WSDL Workspace

If you import a source or target definition from a WSDL but do not create XML views, you can define the views and ports and the relationship among the views in the WSDL workspace. If you create XML views during the import process, you can edit the XML views, ports, and relationships in the WSDL workspace.

If you import a source or target definition from a WSDL but do not create XML views, you can use the WSDL workspace to create views, modify components, add columns, and maintain view relationships in the workspace. When you update a source or target definition, the Designer propagates the changes to any mapping that includes the source or target.

To view or edit a source or target definition in the WSDL workspace, right-click the title of the source definition in the Source Analyzer or the target definition in the Target Designer. Then select the last item in the menu labeled WSDL Workspace.

The WSDL workspace is equivalent to the XML Editor. You use the WSDL workspace the same way you use the XML workspace. However, the WSDL workspace performs validation on changes to the views specific to web service source and target definitions.

Use the following rules and guidelines when you add or edit XML views in web service source or target definitions:

- The source and target definitions for a web service mapping must contain an envelope view equivalent to the SOAP:envelope for the web service request, response, and fault messages.
- A source definition must define views for a output message. It cannot define views for an output or fault message.
- A target definition must define views for an input or fault message. It cannot define views for an output message.
- You can define elements with the type anytype or any. You cannot change the type definition of the soap:Body and soap:Header elements in the source or target definition.
- You can set the default namespace and change the prefix for the namespaces defined in the views of the source or target definition. You cannot change the namespaces.
- You can define new ports as pass-through ports in the envelope view.
- You cannot preview data for the XML views in a web service source or feature definition.

CHAPTER 4

Web Services Consumer Transformation

This chapter includes the following topics:

Working with the Web Services Consumer Transformation Overview

The Web Services Consumer transformation is an active transformation. It performs any function that a web service operation performs. For example, Web Services Consumer transformations can check credit ratings, verify address syntax, send Short Message Service (SMS) messages about workflow status to a cell phone, and handle currency conversion during a PowerCenter session.

You create Web Services Consumer transformations from web service operations. You import a web service operation from a Web Services Description Language (WSDL) file. The Designer imports the request and response elements of the WSDL file.

You can use a WSDL with either a SOAP 1.1 binding or a SOAP 1.2 binding.

Web Services Consumer transformations contain one or more groups and one or more ports. The structure of the transformation depends on how you want to process the operation imported from the WSDL file.

The Designer creates primary and foreign keys to indicate the relationships between groups.

Use the following methods to create a Web Services Consumer transformation:

- Import a web service operation from a remote WSDL file located on a URL.
- Import a web service operation from a local WSDL file.

WSSE Security for Web Services Consumer Transformations

You can select a WSSE security type when you create a Web Service Consumer transformation.

The following table shows the available WSSE security types:

WSSE Security Type	Description
None	The PowerCenter Integration Service does not add the WSSE security header to the generated SOAP request. Default is none.
PasswordText	The PowerCenter Integration Service adds security header for WSSE security authentication to the generated SOAP request. Password is stored in the clear text format. You can also provide a base64 encoded hash password.
PasswordDigest	The PowerCenter Integration Service adds security header for WSSE security authentication. Password is stored in a digest form which provides effective protection against replay attacks over the network. The PowerCenter Integration Service also adds NONCE (a random generated value that is valid only once for that specific username token) and CREATED (timestamp of the Username token in UTC time zone format) to the generated SOAP request.

If you select a WSSE security type as PasswordText or PasswordDigest, the Designer adds the ports wsse_Username and wsse_Password to the web service target definition. You can pass values for the user name and password from upstream transformations. You cannot edit the ports wsse_Username and wsse_Password in the XML editor.

When the PowerCenter Integration Service creates a SOAP request to connect to the web service, the PowerCenter Integration Service adds a WSSE security header. The WSSE security header contains the authentication information in the wsse_Username and wsse_Password ports. The format of the user name and password is based on the security type you selected.

If you select the WSSE security type as None and create ports as wsse_Username and wsse_Password, you cannot edit the ports in the XML editor.

Web Services Consumer Transformation Components

A Web Services Consumer transformation contains the following tabs:

- **Transformation.** You can rename the transformation and add a description on the Transformation tab. Web Services Consumer transformations are reusable and non-reusable.
- **Ports.** View ports on the ports tab.
- **Properties.** Configure transformation properties, such as the runtime location.
- **Metadata Extensions.** Create a non-reusable metadata extension to extend the metadata of the Web Services Consumer transformation. Configure the extension name, datatype, precision, and value. You can also promote a metadata extension to be reusable if you want to make it available to all transformations.
- **Web Services Consumer Properties.** Edit the SOAP action values and add pass-through ports.

Important: If you configure a transformation as repeatable and deterministic, ensure that the data is repeatable and deterministic. If you try to recover a session with transformations that do not produce the same data between the session and the recovery, the recovery process can result in corrupted data.

Configuring Web Services Consumer Transformation Properties

Configure transformation properties on the Properties tab.

The following table describes the Web Services Consumer transformation properties:

Option	Description
Runtime Location	Location that contains the DLL or shared library. Default is $PMExtProcDir. Enter a path relative to the PowerCenter Integration Service node that runs the Web Services Consumer session. If this property is blank, the PowerCenter Integration Service uses the environment variable defined on the PowerCenter Integration Service node to locate the DLL or shared library. You must copy all DLLs or shared libraries to the runtime location or to the environment variable defined on the PowerCenter Integration Service node. The PowerCenter Integration Service fails to load the procedure when it cannot locate the DLL, shared library, or a referenced file.
Tracing Level	Amount of detail displayed in the session log for this transformation. Default is Normal.
Is Partitionable	Indicates if you can create multiple partitions in a pipeline that uses this transformation: - No. The transformation cannot be partitioned. The transformation and other transformations in the same pipeline are limited to one partition. - Locally. The transformation can be partitioned, but the PowerCenter Integration Service must run all partitions in the pipeline on the same node. Choose Local when different partitions of the Web Services Consumer transformation must share objects in memory. - Across Grid. The transformation can be partitioned, and the PowerCenter Integration Service can distribute each partition to different nodes. Default is No.
Transformation Scope	Indicates how the PowerCenter Integration Service applies the transformation logic to incoming data: - Row - Transaction - All Input Default is All Input.
Requires Single Thread Per Partition	Indicates if the PowerCenter Integration Service processes each partition at the procedure with one thread. When you enable this option, the procedure code can use thread-specific operations. Default is enabled if you import the Web Services Consumer transformation in the Transformation Developer. If you use another method to the create the transformation, default is disabled.
Output is Deterministic	Indicates whether the transformation generates consistent output data between session runs. You must enable this property to perform recovery on sessions that use this transformation.

Metadata Extension Properties

You can view the SOAP version of the web service operation in the WSDL as an extension name.

The extension name, datatype, precision, and value cannot be edited.

Viewing the SOAP Version

1. In the Transformation Developer, double-click the title bar of a Web Service Consumer transformation.
   The **Edit Transformations** dialog box appears.
2. Click the **Metadata Extensions** tab.
   You can view the SOAP version.

Web Services Consumer Properties

You can edit the SOAP action values and add pass-through ports to a Web Services Consumer transformation. Pass-through ports are columns that pass non-XML data through the Web Services Consumer transformation. After you add pass-through ports, you can open the WSDL Workspace to add reference ports that refer to the pass-through ports.

The following table describes the attributes on the Web Service Consumer Properties tab:

Attribute	Description
Operation Type	Web service operation encoding type. PowerExchange for Web Services supports RPC/encoded and document/literal encoding types.
Original WSDL Location	URL for the WSDL file that contains the web service operation.
Operation Name	Name of the web service operation.
SOAP Action	Valid SOAP action defined in the WSDL file. This is required for HTTP binding of a web service.
End Point URL	Endpoint URL for the web service host that you want to access. You can use a mapping parameter or variable as the endpoint URL. For example, you can use mapping parameter "$$MyURL" as the endpoint URL and set $$MyURL to the URL in the parameter file.
WS Security Type	Type of WSSE security you want the PowerCenter Integration Service to use. Choose from the following options: - None - PasswordText - PasswordDigest Default is None.

To add a pass through port:

1. On the Web Service Consumer Properties tab, click Add Pass-Through Port.

   The Pass-Through Port dialog box appears.
2. Click the Add button to add an output pass-through port.

   A default field appears in the Field Name column.
3. Modify the field name.
4. Optionally, modify the datatype, precision, and scale.
5. Click OK to close the Pass-Through Port dialog box.
6. Click OK.

Importing a Web Services Consumer Transformation

You can import a Web Services Consumer transformation from a remote or local WSDL file. When you import a Web Services Consumer transformation, you can select a web service operation from a WSDL file located on a URL or a local WSDL file.

You can import definitions from a WSDL file with RPC/Encoded or Document/Literal styles. You can also import definitions from a WSDL file over an HTTPS connection. Web Services Consumer Transformation supports TLS 1.2, TLS 1.1, and TLS 1.0 protocols. The Designer can import Web Services Consumer transformations from nested WSDL file definitions where the WSDL file contains information to import other WSDL files.

When you import a WSDL file and create a Web Services Consumer transformation, you define the structure of the transformation. You can import the transformation with one of the following relationships:

- **Entity relationships**. Creates groups for multiple-occurring or referenced elements and complex types. It creates relationships between views instead of creating one large hierarchy. When you import a transformation with entity relationships, the Designer creates multiple groups.
- **Hierarchical relationships**. Creates a root and expands the XML components under the root. If you create a hierarchical relationship, then you create a normalized view. In a normalized view, every element or attribute appears once. One-to-many relationships become separate XML views with keys to relate the views.

When you import a WSDL file and create a transformation, you can create one or both of the following ports:

- **Cookie port**. Creates a cookie port that can accept cookies and pass them to subsequent PowerCenter Integration Service calls. The PowerCenter Integration Service uses the cookie to authenticate subsequent calls in other Web Services Consumer transformations. Create a cookie port when a remote web server implements user sessions based on cookies.
- **URL port**. Creates a URL port that can receive information about authenticated URLs and pass them to subsequent PowerCenter Integration Service calls. The PowerCenter Integration Service uses the information to authenticate subsequent calls in other Web Services Consumer transformations. Create a URL port when you want to pass a dynamically generated endpoint URL instead of a static endpoint URL to the PowerCenter Integration Service. The value in this column overrides the endpoint URL that you specify in the transformation properties.

To create a Web Services Consumer transformation:

1. Open the appropriate Designer tool.

   Note: If you use the Mapping Designer or Mapplet Designer, you must create a mapping or a mapplet before you create a Web Services Consumer transformation.
2. Click Transformation > Create. Or, click the Web Services Consumer Transformation icon in the toolbar. Then click in the workspace to open the Import from WSDL (Web Services Consumer) dialog box. You can also use this method to replace a transformation. Go to step 6.
3. Select Web Services Consumer as the transformation type.
4. Enter a name for the transformation and click Create.

   The Import from WSDL (Web Services Consumer) dialog box appears.
5. Click Advanced Options to configure the default precision for String datatype fields and to set column naming conventions.

   The Change XMLViews Creation and Naming Options dialog box appears.

You can select the following options:

Option	Description
Override all infinite lengths	You can specify a default length for fields with undefined lengths, such as strings.
Generate names for XML columns	You can choose to name XML columns with a sequence of numbers or with the element or attribute name from the schema. If you use names, choose from the following options: - When the XMLColumn refers to an attribute, prefix it with the element name. PowerCenter uses the following format for the name of the XML column: `NameOfElement_NameOfAttribute` - Prefix the XML view name for every XML column. PowerCenter uses the following format for the name of the XML column: `NameOfView_NameOfElement` - Prefix the XML view name for every foreign-key column. PowerCenter uses the following format for the name of a generated foreign key column: `FK_NameOfView_NameOfParentView_NameOfPKColumn` Maximum length for a column name is 80 characters. PowerCenter truncates column names longer than 80 characters. If a column name is not unique, PowerCenter adds a numeric suffix to keep the name unique.

After you configure these options, the Designer applies them to all Web Services Consumer transformation you create.

6. Click URL to import from a remote WSDL file located on a URL. Or, click Local File to import from a local WSDL file.

   Or, click UDDI to import from a remote WSDL file listed in a UDDI directory. Reserved for future use.
7. If you are importing from a remote WSDL file located on a URL, enter the URL in the Address field. Or, if you are importing from a local WSDL file, navigate to the directory that contains the WSDL file and select the WSDL file.
8. Click Open.

   If you did not configure the advanced settings in step 5, the Designer asks if you want to override the infinite length option. Click Yes to open the Change XMLViews Creation and Naming Options dialog box to configure advanced options.

   Step 1 of the Web Services Wizard appears.
9. Select the web service operation that you want to import and click Next.

   You can only import web service *operations* from a WSDL file. The Import from WSDL (Web Services Consumer) dialog box displays the WSDL definition hierarchy so that you know which port and binding a web service operation is associated with.

   A web service operation might be available to import from more than one port within the same WSDL definition. Make sure that you import the web service operation from the port you want.

   Step 2 of the Web Services Wizard appears.
10. Choose whether you want to generate the XML views as entity relationships or as normalized hierarchical relationships.

    Note: If you select Hierarchy Relationships, then Normalized XML Views is automatically selected. The Denormalized XML Views option is reserved for future use.
11. Optionally, select Create Cookie Port to create a cookie port for the transformation.
12. Optionally, select Create URL Port to create a URL port for the transformation.
13. Click Finish.

Adding Reference Ports

You can add reference ports that refer to pass-through ports that you added to Web Services Consumer transformations. You can add reference ports for request and response components. All other components in the WSDL Workspace are read-only.

To add a reference port:

1. Right-click the title bar of a Web Services Consumer transformation.
2. To edit request components, click WSDL Workspace > Input Mode. Or, to edit response components, click WSDL Workspace > Output Mode.
3. Right-click the top of the group with the pass-through ports in the transformation.
4. Select Add a Reference Port.

   The Reference Port dialog box displays the pass-through ports in the transformation.
5. Select the pass-through port to add to the view and click OK.

   The corresponding output reference port appears in the view. You can rename the port in the Columns dialog box.
6. Click Apply Changes and close the WSDL Workspace.
7. Click OK.

   The WSDL Workspace uses the XML Editor.

CHAPTER 5

Creating and Configuring Web Service Workflows

This chapter includes the following topics:

Working with Web Service Consumer Workflows

When you configure a Web Services Consumer workflow, you define session and scheduler properties that determine how the PowerCenter Integration Service reads data from a web service source, writes data to a web service target, or transforms data using a Web Services Consumer transformation.

You can configure pipeline partitioning for Web Services Consumer sessions. The PowerCenter Integration Service also captures SOAP fault messages to determine the cause of a session failure when you run a workflow.

Pipeline Partitioning

You can increase the number of partitions in a pipeline to improve session performance. Increasing the number of partitions allows the PowerCenter Integration Service to create multiple connections to sources and process source and target partitions concurrently.

The following table describes the partition types for partition points in Web Services Consumer mappings:

Partition Point	Partition Type
Application Source Qualifier for web service sources	Pass-through
Web service target	Pass-through
Web Services Consumer transformation	Pass-through

SOAP Fault Messages

If there is a problem with the SOAP request, the Web Service application connection, or the web services host for the web service you want to access, the session might fail when reading from a web service source, writing to a web service target, or using a Web Services Consumer transformation. The PowerCenter Integration Service captures SOAP fault messages. You can configure the session properties to fail a session when there is a SOAP fault message or to write the message to a target.

SOAP fault messages contain specific information about the error.

The following table describes the SOAP 1.1 fault message elements:

Element	Description
faultcode	Faultcode element can contain any of the following indicators: - VersionMismatch. Invalid namespace for the SOAP Envelope element. - MustUnderstand. Immediate child element of the SOAP header contains the MustUnderstand parameter value set to true. The web services host does not understand the SOAP header and stops processing the rest of the message. - Client. The PowerCenter Integration Service SOAP request contains incorrect data or formatting. - Server. There is a problem with the web services host.
faultstring	Describes the error.
faultactor	Optional. URI identifying the address of the web services host that generated the error.
detail	Optional. Describes the cause of the error.

The following is an example of a SOAP 1.1 fault message:

```
<env:Body>
  <env:Fault>
    <faultcode>env:Client</faultcode>
    <faultstring>Invalid input</faultstring>
    <faultactor>Optional URI</faultactor>
    <detail>Optional additional information</detail>
  </env:Fault>
</env:Body>
```

The following table describes the SOAP 1.2 fault message elements:

Element	Description
Code	A fault identification. The Value element of Code must be one of the following values: - infasoapns:DataEncodingUnknown - infasoapns:MustUnderstand - infasoapns:Receiver - infasoapns:Sender - infasoapns:VersionMismatch
Reason	An explanation of the error.
Node	Contains the URI of the SOAP node that generated the fault.

Element	Description
Role	Optional information about the object that caused the fault to occur.
Detail	Optional information that varies based on the fault.

The following is an example of a SOAP 1.2 fault message:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
   <soapenv:Body>
      <soapenv:Fault>
         <soapenv:Code>
             <soapenv:Value>soapenv:Receiver</soapenv:Value>
         </soapenv:Code>
         <soapenv:Reason>
             <soapenv:Text xml:lang="en-US">Your name is required.</soapenv:Text>
         </soapenv:Reason>
         <soapenv:Detail/>
      </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

If the SOAP fault message indicates that the web service consumer caused the error, you might be able to correct the error and run the session successfully. When a session fails, check the session log. If it contains a SOAP fault message, you can determine if the PowerCenter Integration Service SOAP request caused the error. If the SOAP request caused the error, the faultcode specifies Client. If the faultcode specifies Client, read the SOAP fault message to determine how to modify the SOAP request before running the session again.

Tip: A SOAP fault message cannot identify an improper Web Service application connection or endpoint URL. If you see a SOAP fault message after a session fails, check the application connection settings or the endpoint URL contained in the WSDL file.

If the faultcode specifies Server, the web services host caused the error. If the web services host caused the error, you cannot fix the error. You can only run the session again to see if the server is ready to accept the SOAP request.

Configuring a Session with a Web Services Consumer Mapping

When you configure PowerExchange for Web Services workflows, you can configure web service source, target, and transformation session properties.

You can configure the following session properties:

- **Web Service application connection information.** Optionally, specify a Web Service application connection for web service sources, targets, and Web Services Consumer transformations. If you do not specify a Web Service application connection, or if the Web Service application connection does not have an endpoint URL, the PowerCenter Integration Service uses the endpoint URL contained in the WSDL file. The endpoint URL also appears in the Web Services Consumer Properties when you edit a transformation.
- **Empty XML element handling.** Specify whether to treat empty XML elements as null characters.

- **SOAP request data.** Override the SOAP request that the PowerCenter Integration Service generated for the web service source.
- **SOAP action.** Override the SOAP action value specified in the web service source or target definition or Web Services Consumer transformation.
- **Date-time format.** Choose the local time or Greenwich Mean Time time format for date-time values that the session passes to a web service target or Web Services Consumer transformation.
- **Duplicate parent row handling.** Specify whether the PowerCenter Integration Service fails the session or passes the first or last row when there are duplicate parent rows.
- **Orphan row handling.** Specify whether the PowerCenter Integration Service logs an error or ignores orphan rows.
- **Reset and restart.** You can reset the generated key sequence values or restart them at 1 for sources and transformations at the end of a session.
- **SOAP request cache directory and cache size.** Specify the storage directory for and initial size of intermediate files for targets and transformations created during SOAP request generation.
- **SOAP fault handling.** If a SOAP fault occurs, you can configure the session to fail or to write the fault message to a target.
- **SOAP response compression.** You can configure SOAP response compression for web service sources and targets and Web Service Consumer transformations.

To configure session properties for PowerExchange for Web Services:

1. Double-click a PowerExchange for Web Services session to open the session properties.
2. Click the Mapping tab.
3. If the mapping includes a web service source definition, click the Sources node.
4. From the Connections settings on the Mapping tab (Sources node), optionally select a connection value for Application Multi-Group Source Qualifiers for web service sources.
5. From the Properties settings, optionally modify the following properties:

Property	Description
Treat Empty Content as Null	Treat empty XML elements as null characters.
Reset	Reset the value of generated key sequence values for sources and transformations at the end of a session.
Restart	Restart generated key sequence values for sources and transformations at 1.
SOAP Action	Enter the URI for another web service operation to override the URI for the web service operation you imported for this source definition.
Treat Fault as Error	The PowerCenter Integration Service treats a SOAP fault as a row error and continues the session. When this property is disabled, the fault is sent as output to the data pipeline. Default is enabled.

Property	Description
SOAP Request	Enter a modified SOAP request to override the SOAP request that the PowerCenter Integration Service sends to the web service source. You can view the SOAP request in the Edit Tables dialog box for the web service source definition.
SOAP Response Compression	Select All if you want the PowerCenter Integration Service to enable HTTP response compression. If you select None, the PowerCenter Integration Service uses no compression. If the PowerCenter Integration Service connects to an IIS web server, set SOAP Response Compression to None.

6. If the mapping includes a web service target definition, click the Targets node.
7. From the Connections settings, optionally select a connection value for web service targets.
8. From the Properties settings, optionally modify the following properties:

Property	Description
XML DateTime Format	Choose one of the following date-time formats for data that the session passes to a web service target or Web Service transformation: - Local Time. The time according to the PowerCenter Integration Service server time zone. - Local Time with Time Zone. The difference in hours between the PowerCenter Integration Service time zone and Greenwich Mean Time. - UTC. Greenwich Mean Time.
Duplicate Parent Row Handling	Choose one of the following values to indicate how the PowerCenter Integration Service handles duplicate parent rows during a session: - First Row. The PowerCenter Integration Service passes the first duplicate row to the target. The PowerCenter Integration Service rejects rows with the same primary key that it processes after this row. - Last Row. The PowerCenter Integration Service passes the last duplicate row to the target. - Error. The PowerCenter Integration Service passes the first row to the target. Rows that follow with duplicate primary keys increment the error count. The session fails when the error count exceeds the error threshold.
Orphan Row Handling	Choose one of the following values to indicate how the PowerCenter Integration Service handles orphan rows during a session: - Ignore. The PowerCenter Integration Service ignores orphan rows. - Error. The session fails when the error count exceeds the error threshold.
SOAP Action	Enter the URI for another web service operation to override the URI for the web service operation that you imported for this target definition.
SOAP Request Cache Dir	Directory that contains the temporary files that the PowerCenter Integration Service creates to generate a SOAP request.
SOAP Request Cache Size	Size, in bytes, of the directory that contains the files that the PowerCenter Integration Service creates to generate a SOAP request.
Treat Fault as Error	The PowerCenter Integration Service treats a SOAP fault as a row error and continues the session. When this property is disabled, the fault is sent as output to the data pipeline. Default is enabled.

Property	Description
Null Content Representation	Choose how to represent null content in the target. - No Tag. Do not output a tag. - Tag with Empty Content. Output just the tag. Default is No Tag.
Empty String Content Representation	Choose how to represent null content in the target. - No Tag. Do not output a tag. - Tag with Empty Content. Output just the tag. Default is Tag with Empty Content.
SOAP Response Compression	Select All if you want the PowerCenter Integration Service to enable HTTP response compression. If you select None, the PowerCenter Integration Service uses no compression. If the PowerCenter Integration Service connects to an IIS web server, set SOAP Response Compression to None.

9. If the mapping includes a Web Services Consumer transformation, click the transformation under the Transformations node.
10. From the Connections settings, optionally select a connection value.
11. From the Properties settings, optionally modify the properties.

Property	Description
Treat Empty Content as NULL	Treat empty XML elements as null characters.
Reset	Reset the value of generated key sequence values for sources and transformations at the end of a session.
Restart	Restart generated key sequence values for sources and transformations at 1.
SOAP Action	Enter the URI for another web service operation to override the URI for the web service operation you imported for this source definition.
SOAP Request Cache Dir	Directory where intermediate files are stored for targets and transformations during SOAP request generation.
SOAP Request Cache Size	Initial size of intermediate files created for SOAP request generation.
Treat Fault as Error	The PowerCenter Integration Service treats a SOAP fault as a row error and continues the session. When this property is disabled, the fault is sent as output to the data pipeline. Default is enabled.
Null Content Representation	Choose how to represent null content in the target. - No Tag. Do not output a tag. - Tag with Empty Content. Output just the tag. Default is No Tag.

Property	Description
Empty String Content Representation	Choose how to represent null content in the target. - No Tag. Do not output a tag. - Tag with Empty Content. Output just the tag. Default is Tag with Empty Content.
SOAP Response Compression	Select All if you want the PowerCenter Integration Service to enable HTTP response compression. If you select None, the PowerCenter Integration Service uses no compression. If the PowerCenter Integration Service connects to an IIS web server, set SOAP Response Compression to None.

12. Click OK.

Troubleshooting Web Service Consumer Workflows

I ran a session that contains a web service source or target definition or a Web Services Consumer transformation in its mapping, and the session failed. I see a SOAP fault message in the session log that does not describe the error.

A SOAP fault message cannot identify an improper Web Service application connection. If you see an unspecific SOAP fault message in the session log after a session fails, check the application connection settings or the endpoint URL contained in the WSDL file. Use the Workflow Manager to check the application connection. Check the following application connection characteristics:

- **Name.** Make sure that this is the Web Service application connection that you configured for the web service you want to access during this session.
- **Settings.** Make sure that the settings for the Web Service application connection, such as the endpoint URL, are correct.

If the application connection settings are correct, use a third-party SOAP diagnostic tool to validate the SOAP request.

APPENDIX A

Datatype Reference

This appendix includes the following topic:

Web Services and Transformation Datatypes

PowerCenter uses the following datatypes in PowerExchange for Web Services mappings:

- **XML datatypes.** XML datatypes display in web service source and target definitions in a mapping.
- **PowerCenter transformation datatypes.** PowerCenter transformation datatypes are generic datatypes that PowerCenter uses during the transformation process. They appear in all transformations in a mapping.

Web services communicate using SOAP messages, which contain XML data. XML datatypes are the native datatypes for web services. PowerCenter supports all XML datatypes specified in the World Wide Web Consortium (W3C) May 2, 2001 Recommendation. For more information about the W3C specifications for XML datatypes, see the *XML Schema Part 2: Datatypes* document at http://www.w3.org/TR/xmlschema-2/.

Web Service Source and Target Definitions

Web service source and target definitions in a mapping display XML datatypes. When the PowerCenter Integration Service reads data from a web service source, it converts the native XML datatypes into PowerCenter transformation datatypes used in the Application Multi-Group Source Qualifier transformation. When writing data to a web service target, the PowerCenter Integration Service converts the PowerCenter transformation datatypes into native XML datatypes for the web service target.

Web Services Consumer Transformations

When you create a Web Services Consumer transformation, the Designer converts the native XML datatypes into PowerCenter transformation datatypes. A Web Services Consumer transformation in a mapping displays PowerCenter transformation datatypes.

XML and PowerCenter Transformation Datatypes

For information about XML datatypes and the relationship between XML datatypes and PowerCenter transformation datatypes, see the *PowerCenter XML Guide*.

INDEX

W

X