Informatica

Informatica® Cloud Data Integration for PowerCenter

April 2024

Web Services Provider Guide

Informatica Cloud Data Integration for PowerCenter Web Services Provider Guide
April 2024

© Copyright Informatica LLC 2023, 2024

Publication Date: 2024-04-21

Table of Contents

Preface 7

Chapter 1: Web Service Concepts 8

Web Service Concepts Overview 8

Simple Object Access Protocol (SOAP) 9

Web Services Description Language (WSDL) 9

Chapter 2: Understanding the Web Services Provider 11

Understanding the Web Services Provider Overview 11

- Web Services Hub 11
- Real-time Web Services 12

Web Services Provider Architecture 12

Performance and Scalability 13

- Associating Multiple Repositories with a Web Services Hub 13
- Associating a Repository with Multiple Web Services Hub Services 13
- Running Multiple Instances of a Web Service Workflow 14
- Running Web Service Sessions or Workflows on a Grid 14

Web Services Hub Security 14

Web Services Hub Logs 15

- Configuring the Logs 15
- Viewing the Logs 15

SOAP Fault Handling 15

- User-Defined Faults 15
- System Faults 16
- System Fault Schema 16

Chapter 3: Using the Web Services Hub Console 18

Using the Web Services Hub Console Overview 18

Connecting to the Web Services Hub Console 18

Understanding the Web Services Hub Console 19

- Navigator 19
- Web Services Sections 19
- Properties Section 20

Testing a Web Service 21

- Input Message 21
- Testing a Public Web Service Operation 22
- Testing a Protected Real-time Web Service 23

Chapter 4: Writing Client Applications 24

Writing Client Applications Overview 24

Client Applications for Real-time Web Services. 24
Web Service Workflows. 25
Generating Client Proxy Classes. 25
Initialization. 25
Operation Calls. 25
Error Handling. 25
Java Client Application for Real-time Web Services. 25
Creating a Client Application for a Real-time Web Service. 26
Using Parameter Arrays. 28
Parameter Array Definition. 28
Rules and Guidelines for Using Parameter Arrays. 29
Adding Security to a Client Request. 30
UsernameToken in the SOAP Request. 30
Plain Text Password. 31
Hashed Password. 31
Digested Password. 33

Chapter 5: Working with Web Service Sources and Targets. 35

Web Service Sources and Targets Overview. 35
Understanding Web Service Sources and Targets. 36
XML Views and Groups. 36
Source Definition. 36
Target Definition. 37
Rules and Guidelines for Importing or Creating Web Service Sources and Targets. 37
Importing a Web Service Source or Target Definition. 38
Import Modes. 39
Message ID. 39
Advanced Options. 39
Importing from a WSDL Without Creating XML Views. 40
Importing a Web Service Source or Target Definition from a WSDL. 40
Creating a Source or Target Definition. 41
Multiple Occurring Elements. 42
Message Ports. 42
Creating a Source or Target from a Relational or Flat File Source or Target. 43

Chapter 6: Editing Web Service Sources and Targets. 45

Editing Web Service Sources and Targets Overview. 45
Editing Definitions in the Designer Workspace. 45
Table Tab. 46
Columns Tab. 46
Attributes Tab. 46
Web Service Definition Tab. 46
Editing Definitions in the WSDL Workspace. 46

Rules and Guidelines for the WSDL Workspace. 47

Chapter 7: Working with Web Service Mappings. 49

Working With Web Service Mappings Overview. 49
Types of Web Service Mappings. 50
Request-Response Mappings. 50
Staged Mappings. 51
Generating a Mapping from a WSDL. 51
Generating a Mapping from a Relational or Flat File Source or Target. 52
Generating a Mapping from a Transformation or Mapplet. 53
Generating a Mapping from a Reusable Transformation. 53
Generating a Mapping from a Mapplet. 53
Generating a Mapping from a Reusable Transformation or a Mapplet. 54
Editing a Target Instance in a Web Service Mapping. 54
Load Scope. 55
Partial Load Recovery. 55
Attachments. 55
Flat File or XML Source and Target Attachments. 55
WSDL Attachments. 56

Chapter 8: Working with Web Service Workflows. 57

Working with Web Service Workflows Overview. 57
Creating and Configuring a Web Service Workflow. 58
Creating a Web Service Workflow. 58
Configuring the Web Service Workflow. 58
Concurrent Execution of Web Service Workflows. 60
Configuring the Web Services Provider Reader and Writer. 61
Configuring the Web Services Provider Reader. 61
Configuring the Web Services Provider Writer. 62
Configuring the Reader and Writer for XML and Flat File Sessions. 64
Configuring Partitions for Web Service Sessions. 64
Troubleshooting Web Service Workflows. 65

Appendix A: Web Service Sample Client Applications. 67

Web Service Sample Client Applications Overview. 67
Using the Real-time Web Services Sample Programs. 67
Step 1. Create the Lookup Tables. 68
Step 2. Import the Mappings and Workflows. 69
Step 3. Modify the Database and Datatypes for the SQL Transformation 69
Step 4. Modify the Database Connection Settings. 70
Step 5. Compile the Real-time Web Service Sample Programs. 70
Step 6. Run the Real-time Web Service Sample Programs. 70
Examples of Real-time Web Services. 70

Multiple Row Lookup. 71
Single Row Lookup. 71

Appendix B: Configure the Web Browser. 73

Configure the Web Browser. 73

Index. 74

Preface

See the *Cloud Data Integration for PowerCenter (CDI-PC)Web Services Provider Guide* for information about the Web Services Provider and the CDI-PC web services hosted by the Web Services Hub. Use the guide to learn how to turn CDI-PC workflows into web services and to see examples that create client applications that use the web services available on the Web Services Hub.

CHAPTER 1

Web Service Concepts

This chapter includes the following topics:

- Web Service Concepts Overview, 8
- Simple Object Access Protocol (SOAP), 9
- Web Services Description Language (WSDL), 9

Web Service Concepts Overview

Web services are business functions that operate over the Web. They describe a collection of operations that are network accessible through standardized XML messaging. You can write applications that can communicate with Integration Services in any language or platform. You can embed these applications easily in existing components and products.

Web services are based on open standards, such as XML, SOAP, and WSDL, which offer greater interoperability than traditional proprietary applications.

Examples of web services include business services, such as stock quotes, airline schedules, and credit checks.

The components that enable web services include:

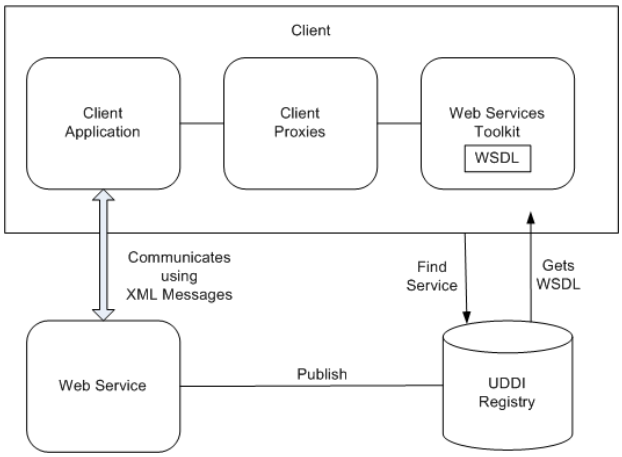
- **Simple Object Access Protocol (SOAP).** SOAP is the communications protocol for web services. It is the specification that defines the XML format for web service messages.
- **Web Service Definition Language (WSDL).** WSDL is an XML document that describes web service operations.
- **Registry.** Directory of published web services. Some web service providers publish services in Universal Description, Discovery, and Integration (UDDI). Registering a web service in the UDDI is optional.

  Note: The CDI-PC Web Services Provider does not use the UDDI registry.

To build a web service client for the CDI-PC Web Services Provider, you select the web service you want to interface with and retrieve the WSDL for the selected web service. Use a web service tool kit such as Axis to generate the client proxies. The client proxies contain all of the function calls required to interact with a web service.

You can determine what functions a web service offers, the data the web service requires, and the location of the service by examining the WSDL. The WSDL describes the web service interfaces and the operations available for the service. Use the information in the WSDL to build a client application to use the services.

The following figure shows the building blocks of a web service:



Simple Object Access Protocol (SOAP)

SOAP is the communications protocol for web services. It defines the format for web services messages. SOAP Encoding is used to tell the SOAP runtime environment how to translate from data structures, such as Java, into SOAP XML. SOAP and the WSDL dictate the communication between web services and their clients.

A SOAP message contains the following sections:

- **SOAP envelope.** The envelope defines the framework of the message, including the content of the message, who or what should handle it, and whether it is optional or mandatory.
- **SOAP header.** The header is an element of the SOAP envelope that lets you add features to a SOAP message in a decentralized manner.
- **SOAP body.** The body is the container for mandatory information that provides a mechanism for exchanging information with the intended recipient.

Authentication and transaction management are typical examples of extensions that can be implemented as header entries. The SOAP header helps to process the data in the body of the SOAP message. Information related to authentication or transactions is usually contained in the header because this information identifies the entity that sent the SOAP message body and the context in which it will be processed.

Use a SOAP toolkit to create and parse SOAP messages. A SOAP toolkit translates function calls from another language to a SOAP message. For example, the Apache Axis toolkit translates Java function calls to SOAP.

Use SOAP to implement web services on different platforms both inside and outside an organization. Each SOAP implementation supports different function calls and parameters. Therefore, a function that works with one toolkit may not work with another.

Web Services Description Language (WSDL)

The WSDL is an XML document that describes the protocols and formats used by a web service.

The WSDL contains a description of the data to be passed to the web service so that both the sender and the receiver of the service request understand the data being exchanged. The WSDL elements also contain a description of the operations to be performed on that data, so that the receiver of a message knows how to process it, and a binding to a protocol or transport, so that the sender knows how to send it.

You can view and download the WSDL files for the web services hosted by the CDI-PC Web Services Provider on the Web Services Hub Console.

Related Topics:

- "Using the Web Services Hub Console" on page 18

CHAPTER 2

Understanding the Web Services Provider

This chapter includes the following topics:

- Understanding the Web Services Provider Overview, 11
- Web Services Provider Architecture, 12
- Performance and Scalability, 13
- Web Services Hub Security, 14
- Web Services Hub Logs, 15
- SOAP Fault Handling, 15

Understanding the Web Services Provider Overview

The Web Services Provider consists of the following components:

- **Web Services Hub.** An application service in the CDI-PC domain that uses the SOAP standard to receive requests and send responses to web service clients. The Web Services Hub interacts with the Integration Service and the Repository Service to process requests and generate responses.
- **Real-time web services.** When you enable CDI-PC workflows as web services, you create real-time web services. When you transform CDI-PC workflows into web services, you can run the workflows from web service clients.

Web Services Hub

The Web Services Hub is the web service gateway in the CDI-PC domain that allows client applications access to CDI-PC functionality using web service standards and protocols. With the Web Services Hub, you can enable CDI-PC workflows as web services. You can also monitor CDI-PC processes and get repository information.

The Web Services Hub allows the data integration processes to remain within the CDI-PC framework, but handles requests and responses using web service technologies. The Web Services Hub receives requests from web service clients in the form of SOAP messages and passes them to the Integration Service. The Integration Service works with the Repository Service to process the requests and sends the results to the Web Services Hub. The Web Services Hub sends a response back to the web service client in the form of SOAP messages.

The Web Services Hub provides a Web Services Hub Console where you can manage web services and view and download the WSDL files for the web services. You can use the WSDL files to create client applications to access the web services.

The CDI-PC installation includes the Web Services Hub. After you install CDI-PC, use Informatica Administrator to create a Web Services Hub and enable it as you would other application services in the domain.

Real-time Web Services

When you initially start the Web Services Hub after installation, no real-time web services are available. You create real-time web services when you expose CDI-PC workflows as web services. You can create clients to run a web service workflow and get the results of the workflow process. The web service takes a SOAP message request and produces a SOAP message response.

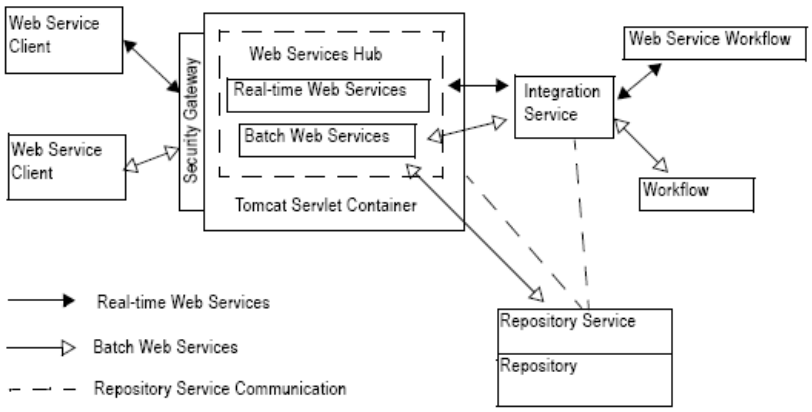
You can create a service mapping to receive a message from a web service client, transform it, and write it to any target that CDI-PC supports. You can also create a web service mapping with both a web service source and target definition to receive a message request from a web service client, transform the data, and send the response back to the web service client. The source and target definitions represent service operations. The source defines the user request and the target defines the response.

After you create a mapping, you can create a web service workflow to run the process defined in the web service mapping. A web service workflow is a workflow enabled as a web service. Configure the web service workflow, and add sessions to the workflow. When you save the workflow, the Web Services Hub publishes the web service on the Web Services Hub Console. The Integration Service can perform parallel processing of both request-response and one-way services.

Web Services Provider Architecture

The Web Services Provider consists of the Web Services Hub and real-time web services hosted by the Web Services Hub. The Web Services Hub works with the Integration Service and the Repository Service to process web service requests.

The following figure shows the Web Services Provider architecture:



The following process describes how the Web Services Hub processes web service requests:

1. A web service client sends a SOAP message to the Web Services Hub to run a web service.

2. For protected real-time web services, the Web Services Hub authenticates the web service client based on the user name token.
3. The Web Services Hub generates a message ID for the request.

   If the request is for a real-time web service, the Web Services Hub sends the message to the Integration Service.
4. The Integration Service or Repository Service processes the request.

   If the request is for a real-time web service, the Integration Service sends the processed data to the Web Services Hub which uses the message ID to correlate the request with the response.
5. The Web Services Hub sends a SOAP response to the web service client.

The Integration Service and Web Services Hub communicate with the Repository Service throughout the process.

Performance and Scalability

You can run more than one Web Services Hub on a single node. When you run multiple Web Services Hub on one node, you increase the number of web services that you can run on a node and maximize the use of your resources.

When you configure the Web Services Hub to run web services in a domain, you can use the following options to improve performance and provide flexibility and scalability:

- Associate multiple repositories with a Web Services Hub.
- Associate a repository with multiple Web Services Hubs.
- Run multiple instances of a web service workflow.
- Run web service sessions or workflows on a grid.

Associating Multiple Repositories with a Web Services Hub

You can associate more than one repository with a Web Services Hub. When you associate multiple repositories with a Web Services Hub, the Web Services Hub can run web service workflows located in any of the associated repositories. This allows you to use one Web Services Hub to run web services that might be accessed by different users at different times, which maximizes the use of the Web Services Hub.

Associating a Repository with Multiple Web Services Hub Services

You can associate a repository with more than one Web Services Hub. When you associate one repository with multiple Web Services Hub Services, multiple Web Services Hub Services can run the same web services.

Use a third party load balancer to manage and distribute requests to the Web Services Hub Services so that the service request load is balanced across the Web Services Hub Services. A hardware load balancer used in a production environment can optimize the performance of web services. Set the URL for the load balancer when you create a Web Services Hub Service in the Administrator tool.

Running Multiple Instances of a Web Service Workflow

If you configure a workflow to run in more than one instance, the Web Services Hub can dynamically start new instances of the web service to handle as many web service requests as possible. The Web Services Hub monitors web service usage to determine resource usage and web service processing times. You can set a threshold for the maximum time the Web Services Hub can take to process requests for a web service. When processing time exceeds the threshold, the Web Services Hub starts another instance of the web service workflow to process new requests.

When the number of service requests decreases, the Web Services Hub can dynamically shut down web service instances to reduce resource usage.

Related Topics:

- "Concurrent Execution of Web Service Workflows" on page 60

Running Web Service Sessions or Workflows on a Grid

When a CDI-PC domain contains a grid, you can run a web service workflow on a grid. Create the grid and associate an Integration Service with the grid in the Administrator tool. Then assign the Integration Service to run the web service workflow.

To run a web service workflow on a grid from a client application, run the web service workflow on the Integration Service associated with a grid.

You can also enable the session to run on a grid. When a session runs on a grid, the Integration Service distributes the session threads across the nodes in a grid. To run a session on a grid, add a message ID to the web service source and target definitions. The Integration Service uses the message ID to associate the web service input and output messages across the nodes.

Web Services Hub Security

The Web Services Hub has the following levels of security:

- **Encryption.** The Web Services Hub encrypts the repository login information in the configuration file used to connect to the repository. You can also run the Web Services Hub in secure mode and use the SSL protocol for encryption of web service client requests.
- **Authentication.**

  For protected real-time web services, the Web Services Hub authenticates the web service client based on the user name token. The web service client must include the user name token in every SOAP request sent to the Web Services Hub. The user name token can include a plain text, hashed, or digested password.

  The Web Services Hub does not authenticate web service requests for a public real-time web service.
- **Authorization.** A web service client with repository access must have permission on a folder to run a service. For protected real-time web services, a web service client with the appropriate permissions on a folder can run a service in that folder based on service configuration. For example, if the service is not runnable, a web service client cannot start the service, but it can invoke the service if the web service workflow is running.

Related Topics:

- "Adding Security to a Client Request" on page 30

Web Services Hub Logs

The Web Services Hub creates a log for status and error messages related to tasks, such as service initialization, task execution, and connection status. The logs include the IP address of the client, the service the client invokes, and the associated workflow. You can troubleshoot problems by examining error messages in this log.

You can view and configure the logs for the Web Services Hub on the Administrator tool.

Note: The Web Services Hub also writes messages in the fault element of a SOAP response when it cannot process the request.

Related Topics:

- "SOAP Fault Handling" on page 15

Configuring the Logs

The Log Manager in the CDI-PC domain handles all logging functions for all services in the domain, including the Web Services Hub.

In the Administrator tool, you can configure the size and location of the Web Services Hub logs and the error level that would be included in the logs.

Viewing the Logs

You can view Web Services Hub log events on the Administration Console Log Viewer. You can filter log events to get a list of only the log events for the Web Services Hub. When you view log events in the Log Viewer, the Log Manager displays the log events from the generated files in the log directory set by the domain administrator.

SOAP Fault Handling

The Web Services Hub sends error responses as SOAP fault messages. The Web Services Hub can generate the following types of fault responses:

- User-defined faults
- System faults

User-Defined Faults

To send error data to the target, you can define fault views in the target definition. If the transformation logic in the web service mapping sends error data to the target, the Integration Service writes messages to fault targets. Send error data to the target when you want to catch and resolve specific errors. For example, you

expect the datatype of the response to be a string. If the web service workflow sends a numeric response, you can send the response to the fault target. You can then evaluate the response and resolve the error.

System Faults

If the Web Services Hub encounters system errors, it generates a fault message based on the type of error and sends the response to the web service client. The fault message is based on the task the Web Services Hub performs when it encounters the error:

- If the Web Services Hub cannot process the header element of a SOAP request message, it returns error information related to the header entries of the SOAP request message in a child element of the SOAP response header element.
- If the Web Services Hub encounters any error with the header element of a SOAP request, it does not process the body element. The SOAP response to the request contains the header fault element in the SOAP header and a SOAP fault element without the detail element.
- If the Web Services Hub cannot process the contents of the body element, the SOAP fault element in the SOAP response message contains a detail element with error information.
- The Web Services Hub generates a SOAP fault response with the error information in the detail element when it encounters any of the following system errors:
  - The Integration Service is not running and the Web Services Hub cannot process the input message.
  - The Web Services Hub has timed out.
  - The protected web service does not provide a valid user name token.
- The Web Services Hub does not return a response for a web service request in the following situations:
  - The content of the service request is malformed or generates a parsing error.
  - The workflow filters out the request.

System Fault Schema

By default, system fault messages contain a message code that includes a prefix and code number and the message text. For example, the message code WSH_95002 is associated with an invalid request that includes an empty workflow name.

The message code is the ErrorCode element in the detail element of a SOAP fault, and the message text is the faultstring element of the SOAP fault.

SOAP Fault Header

The Web Services Hub reports header related errors in the header fault element of a SOAP response header.

The schema of this element is listed below:

```
<ns1:HeaderFault xmlns:ns1=”http://www.informatica.com/wsh”>
   <ErrorCode>
      error code
   </ErrorCode
   <ErrorMessage>
      error message
   </ErrorMessage>
</ns1:HeaderFault>
```

SOAP Fault Body

The SOAP fault body contains the following sub-elements:

- **Faultcode.** The faultcode determines if the error originates at the web service client or the Integration Service. The error can originate at the web service client if the message has the wrong structure.
- **Faultstring.** The faultstring provides a description of the error. The faultstring value indicates whether the error originated from the Integration Service, Web Services Hub, or Repository Service.
- **Detail.** The detail element contains error information that includes an error code, and the extended details element provide detailed error information when the faultstring is a Web Services Hub or Repository Service error.

The Web Services Hub uses the following SOAP fault schema:

```
<SOAP-ENV: Fault>
      <faultcode> Client/Server </faultcode>
      <faultstring>Brief Description of Error</faultstring>
      <detail>
          <ns:WSHFaultDetails xmlns:ns="www.informatica.com/wsh">
              <ErrorCode>
                 Error Code
              </ ErrorCode >
              <ExtendedDetails>
                 Actual Error
              </ ExtendedDetails >
          </ns:WSHFaultDetails>
      </detail>
</SOAP-ENV: Fault>
```

CHAPTER 3

Using the Web Services Hub Console

This chapter includes the following topics:

- Using the Web Services Hub Console Overview, 18
- Connecting to the Web Services Hub Console, 18
- Understanding the Web Services Hub Console, 19
- Testing a Web Service, 21

Using the Web Services Hub Console Overview

The Web Services Hub Console is the CDI-PC application you use to view and test the real-time web services operations available in a Web Services Hub. Use the Web Services Hub Console to perform the following tasks:

- **View the properties of a real-time web service.** You can view the description of the web service and properties such as whether the web service is protected. You can also view the repository and folder that contains the web service.
- **View the WSDL for a real-time web service.** To download the WSDL, save the WSDL to a file on the hard disk.
- **Test a real-time web service.** Use the Try-It client application to run a valid web service and view the response on the Web Services Hub Console.

Note: The Web Services Hub Console does not require authentication. You can access the Web Services Hub Console without logging in. To ensure security, run the Web Services Hub within a secure network environment.

Connecting to the Web Services Hub Console

You can connect to the Web Services Hub Console from any browser.

Use one of the following URLs to connect to the Web Services Hub Console:

```
http://<WebServicesHubHostName:PortNumber>/wsh
http://<WebServicesHubHostName:PortNumber>/CDI-PC
```

The context names /wsh and /CDI-PC are case sensitive.

The default port for a Web Services Hub running on HTTP is 7333. You can also configure the Web Services Hub to use a secure connection with HTTPS. The default port for a Web Services Hub running on HTTPS is 7343. You can set the port number when you create the Web Services Hub in the Administrator tool.

You can also connect to the Web Services Hub Console from the Administrator tool. View the details of the Web Services Hub and click the Service URL. You must enable the Web Services Hub to connect to the Web Services Hub Console.

Understanding the Web Services Hub Console

The Web Services Hub Console consists of the following sections:

- **Navigator.** The Navigator displays the types of services that you can view on the Web Services Hub Console.
- **Web Services or Operations.** For real-time web services, the Web Services section displays valid and invalid web services.

  In the Web Services section, you can test a web service or view the WSDL for a web service.
- **Description.** The Description section provides information on the type of web services selected in the Navigator.
- **Properties.** The Properties section displays the properties of the web service or web service operation selected in the Web Services.

Navigator

In the Navigator, you can scroll and select the type of web service for which you want to display information. The information displayed in the other sections of the console varies based on the type of web service you select in the Navigator.

Web Services Sections

The Web Services Hub Console displays the Web Services section section depending on which type of web service you select in the Navigator.

When you select Valid Web Services or Invalid Web Services in the Navigator, the Web Services section displays information about the real-time web services that run on the Web Services Hub.

You can sort the list of web services or operations. To sort the list of web services, click the label of the column by which you want to sort. The Web Services Hub Console lists the web services alphabetically based on the column you click. An arrow next to the column label shows the sort order for the list, ascending or descending.

You can use the Try-It application to test a web service operation listed in the Web Services. To test a web service operation, enter the values for the parameters in the input message of the web service operation and view the response.

In the Web Services section, you can display the WSDL for a web service. Use the WSDL to write client applications that call the real-time web service.

Web Services Section

You must create web service workflows to view real-time web services on the Web Services Hub Console. When you configure a web service workflow to be visible, the Web Services Hub publishes the web service and WSDL on the Web Services Hub Console.

When you select Valid Web Services or Invalid Web Services in the Navigator, the Web Services section displays the list of real-time web services configured to be visible on the Web Services Hub Console.

If you have privileges to manage objects in a repository, you can view all web services associated with the repository. You can view but cannot run web services created by other users. For example, you have Create, Edit, and Delete privileges on the run-time objects for the TestRepo repository. On the Web Services Hub Console for a Web Services Hub associated with the TestRepo repository, you can view all the web services in the TestRepo repository. You can view but cannot run web services in the TestRepo repository created by other users.

The following table describes the options available in the Web Services section:

Label	Description
Try-It	Client application you can use to test the selected web service. Click to run the selected web service. Not available for invalid web services.
WSDL	WSDL for the selected web service. Click to view the WSDL for the selected web service. You can click the WSDL button at the top of the section or in the same row as the selected web service. To download the WSDL, view and save the WSDL to your local machine. Not available for invalid web services.
Search	Search for web services. Enter the string of text you want to search for and click Go. The Web Services section lists any web service name, repository name, or workflow name that contains the text.
Service Name	Name of the web service that you can run on the Web Services Hub.
Repository Name	Name of the repository associated with the web service.
Workflow Name	Name of the workflow that comprises the web service.

Properties Section

The Properties section displays information about the web service or web service operation selected on the Web Services or Operations section.

Properties Section for Real-time Web Services

When you select a valid or invalid real-time web service in the Web Services section, the Properties section displays the properties of the selected web service.

The following table describes the real-time web service properties:

Property	Description
Service Name	Name of the web service.
Domain Name	Name of the CDI-PC domain that contains the Web Services Hub.
Repository Name	Repository that contains the web service workflow.
Folder name	Name of the folder that contains the web service workflow.
Workflow Name	Name of the workflow associated with the web service.
Description	Description of the web service.
Is Runnable	Indicates whether a web service can be started by a client application. If True, a web service client can start the web service workflow or invoke the web service while the workflow is running. If False, a web service client can invoke the web service while the workflow is running, but cannot start the workflow.
Is Protected	Indicates whether the web service is protected or public. If True, a web service client request must pass authentication. The SOAP request must include a valid user name token in the header. If False, any web service client can run web service requests without authentication.
Is One Way	Indicates whether the web service uses one-way or request-response mapping.

Testing a Web Service

The Try-It application is a client application that you can use to run a real-time operation listed in the Web Services Hub Console. Use the Try-It application to test a valid web service operation and view the results on the Web Services Hub Console. You can use the Try-It application if you are unsure what parameters are required in the input message or you want to view the response for a specific input message.

You can use the Try-It application to run a web service or call an operation without needing to download the WSDL and generate the client proxy classes for a client application. You can view the response on the console and determine how a client application should process the response from the web service.

You can test a valid real-time web service application operation. You cannot use the Try-It application to test a web service with a WSDL that contains a SOAP attachment.

Protected real-time web services require authentication. To test a protected web service operation, provide a valid user name token to log in to the CDI-PC repository.

Input Message

The Try-It application provides two methods for creating a web service request:

- XML input
- Form input

Use the method that best fits the requirements of the request. For example, if the request includes multiple occurring elements, use XML input to create the request message.

XML Input

When you select the XML Input tab, the Web Services Hub displays a SOAP input message that contains the elements needed to run the service request operation. Enter the values for the elements in the SOAP message. Or, you can create a SOAP message outside the Web Services Hub console and paste it into the XML input section.

The Web Services Hub uses the SOAP input message to run the web service. It displays the response as a SOAP output message.

Form Input

When you select the Form Input tab, the Web Services Hub displays a list of the parameters for a web service request. Enter the values for the parameters. If the web service request contains complex type elements, the Form Input tab displays the input parameters in the correct hierarchy.

The Web Services Hub uses the parameter values you enter to create a SOAP input message and run the web service. It displays the response as a SOAP output message.

Testing a Public Web Service Operation

To test a public web service, select a web service operation and enter the values for the parameters in the input message of the web service operation.

To test a public web service operation:

1. In the Web Services section, select a valid real-time web service or an operation.
2. Click Try-It.

   The Try-It application window displays a list of the web service operations that you can test and instructions on how to run the Try-It application.
3. Select the operation you want to test.

   The Try-It application window displays the parameters for the input message.
4. Click the XML Input tab to enter the input parameters in SOAP message format.

   Or, click the Form Input tab to enter the input parameters in a parameter entry form.
5. Enter the values for the parameters.

   The WSDL can contain user-defined datatypes. To avoid fault responses, enter the value for the parameter according to the datatype.
6. Click Send.

   The Web Services Hub runs the web service operation and displays the SOAP message response and a message to indicate success or failure.
7. To clear the parameters and enter new values, click Reset.
8. Click the Close button of the web browser to exit the Try-It application window and return to the main page of the Web Services Hub Console.

Testing a Protected Real-time Web Service

To test a protected real-time web service, include a valid user name token in the SOAP header. You can enter the user name and password in the Form Input tab or modify the SOAP message to include all elements of the user name token in the XML Input tab.

You can test a protected web service with a plain text or hashed password in the Form Input or XML Input tab. To test a protected web service with a hashed password, encrypt the password with the MD5 or SHA-1 hash function before you test the web service. The encryption must be encoded in Base64. Use the resulting hashed value as the password for the web service.

You can test a protected web service with a digested password in the XML Input tab. To test a protected web service with a digested password, add the Password attribute and elements required in the UsernameToken element for digested passwords.

To test a protected web service:

1. In the Web Services section for real-time web services, select the protected web service to run and click Try-It.
2. In the Try-It application window, select the operation for the protected web service.
3. To use the Form Input to test the web service operation, click the Form Input tab.

   In the SOAP header section, enter the user name and a plain text or hashed password.

   In the SOAP body section, enter the values for the parameters required by the protected web service.

   -or-

   To use the XML Input to test the web service operation, click the XML Input tab and update the UsernameToken element.

   To test a protected web service that uses a plain text or hashed password, replace the value *[string]* in the Username and Password child elements with a valid user name and password:

   ```
   <UsernameToken>
       <Username>[string]</Username>
       <Password>[string]</Password>
   </UsernameToken>
   ```

   To test a protected web service that uses a digested password, replace the value *[string]* in the Username element with a valid user name. Update the Password element and add Nonce and Created elements with the appropriate value:

   ```
   <UsernameToken>
       <Username>[string]</Username>
       <Password Type="PasswordDigest">[string]</Password>
       <Nonce>[NonceValue]</Nonce>
       <Created>[RequestCreationTimestamp]</Created>
   </UsernameToken>
   ```

   For more information about the UsernameToken element, see "UsernameToken in the SOAP Request" on page 30.

   In the SOAP body section, enter the values for the parameters required by the protected web service.
4. Click Send.

   The Web Services Hub runs the protected web service operation and displays the SOAP message response on the console.
5. Click the Close button of the web browser to exit the Try-It application window and return to the main page of the Web Services Hub Console.

CHAPTER 4

Writing Client Applications

This chapter includes the following topics:

- Writing Client Applications Overview, 24
- Client Applications for Real-time Web Services, 24
- Java Client Application for Real-time Web Services, 25
- Using Parameter Arrays, 28
- Adding Security to a Client Request, 30

Writing Client Applications Overview

This chapter provides an overview of how you can write client applications to use the web services offered by the CDI-PC Web Services Provider. The general discussion on the steps to create a client application is followed by examples of how to create client applications in the Java and .NET frameworks.

To create a client application for the CDI-PC web services, you need the web service WSDL and a web service toolkit. Web services toolkits make it easy to create client applications by generating client-side proxy classes from the web service WSDL. You can use the Microsoft .NET and Apache Axis web services toolkits to write client applications for the CDI-PC web services.

You can create a client application to run CDI-PC real-time web services. The application development follows the same basic steps.

Note: The Web Services Hub can process chunked messages. To enable chunked transfer encoding in your client request, add the following header to the SOAP message:

```
TRANSFER ENCODING=chunked
```

Client Applications for Real-time Web Services

Client applications for real-time web services involve the following elements:

- Web service workflows
- Client proxy classes
- Initialization
- Operation calls
- Error handling

Web Service Workflows

You build real-time web service client applications to run web services workflows. Before you create the client application, create the mappings and workflows in CDI-PC. Enable the following options in the workflow to allow a client application to run the workflow:

- **Web Service.** Enable the Web Service option to turn a workflow into a web service workflow.
- **Runnable.** Enable the Runnable option to allow a client application to run the web service workflow.
- **Visible.** Enable the Visible option so that the Web Services Hub publishes the WSDL for the web service in the Web Services Hub Console.

Generating Client Proxy Classes

To use real-time web services you create in CDI-PC, you need to generate client proxy classes from the WSDL of the web service you want to access.

To generate client proxies, complete the following steps:

1. Select the web services toolkit for the platform and language in which you want to develop.
2. Download the WSDL for the real-time web service from the Web Services Hub Console.
3. Generate the client-side proxy classes from the WSDL using the web service toolkit. Refer to the web services toolkit documentation for details on generating proxy classes. Each toolkit generates the client proxy classes in a specific way.

Initialization

The client application must instantiate the web service object in the client proxy classes and get the port for the web service before the application can make calls to the web service operations.

Operation Calls

To invoke a web service operation, the client application must create a request object and pass it to the port operation. When the web service sends back a response, the client application must handle the response as needed.

Error Handling

SOAP fault elements in the SOAP response contain the errors that occur during calls to web services. The client application should implement the appropriate error handling scheme to retrieve the SOAP fault.

Java Client Application for Real-time Web Services

This section provides instructions for using the Axis Web Services Toolkit to create a Java client application program that calls a CDI-PC real-time web service. For more information about using the Axis Web Services Toolkit see the documentation on the Apache web site:

```
http://ws.apache.org/axis/java/user-guide.html
```

Before you create the client application that calls a CDI-PC web service workflow, you must first create the web service workflow and generate the WSDL for the web service. You then create the client application based on the web service WSDL.

To create a CDI-PC web service and generate the WSDL, complete the following steps:

1. Create a mapping for the web service workflow. You can create a mapping to receive a message from a web service client, transform the data, and send the response back to the web service client or write it to any target that CDI-PC supports.
2. Create a workflow and enable it as a web service. Create a workflow to run the mapping and enable the Web Services option in the workflow properties. Select the Runnable option so that client applications outside of CDI-PC can run the workflow.
3. Locate and download the WSDL for the web service workflow. When you create the web service workflow, CDI-PC generates a WSDL for the web service. If you configure the web service to be visible, you can view the WSDL on the console of the Web Services Hub associated with the web service.

After you create the web service, you can develop a client application to run the web service workflow.

Related Topics:

- "Creating and Configuring a Web Service Workflow" on page 58
- "Web Services Section " on page 20

Creating a Client Application for a Real-time Web Service

To create a client application that calls a real-time web service, complete the following steps:

1. Generate the client proxy classes for the web service.

   After you create the proxy classes, create the Java application to call the web service. Perform the next steps within the Java application.
2. Initialize the web service objects.
3. Create the request object.
4. Pass the request object to the port operation and handle the response.

Note: The sample code snippets in the following sections are taken from the real-time web services sample program for multiple row lookup. You can view the example in the following directory:

```
<CDI-PCInstallationDir>/server/samples/WebServices/samples/RealTimeWebServices/
UnprotectedWebServices/axis/CustomerLookup_MULTIPLEROW
```

Step 1. Generate Client Proxy Classes in Axis

You can use the Axis Web Services Toolkit to generate Java client proxy classes for the web service WSDL. Specifically, you can run the WSDL2Java tool to generate the Java proxy class files.

Verify that the WSDL has the correct host name and port number for the web service in the endpoint URL. If the endpoint URL is not correct, update the address element, which is available in the \definitions\service \port hierarchy in the WSDL.

Use the following command to generate the client proxy classes:

```
java org.apache.axis.wsdl.WSDL2Java -W <WSDLFile>
```

For example, for WSDL named SampleWS.wsdl, run the following command:

```
java org.apache.axis.wsdl.WSDL2Java -W SampleWS.wsdl
```

The -W option turns off support for wrapped document literal services.

WSDL2Java generates a class for each data type defined in the WSDL. By default, WSDL2Java generates package names based on the namespaces in the WSDL. Typically, if the namespace is of the form *http://x.y.com* or *urn:x.y.com*, the corresponding package will be *com.y.x*.

Step 2. Initialize the Web Service Objects

Before you call any web service operation, you must create the web service object in the client proxy classes and get the port for the web service.

To create the web service object, instantiate the service locator classes. In the sample program, the following code instantiates the service locator:

```
CustomerLookup_MULTIPLEROW service = new CustomerLookup_MULTIPLEROWLocator();
```

To get the port for the web service, use the proxy class created for the port type. In the sample program, the following code gets the port for the web service:

```
CustomerLookup_MULTIPLEROWPort port =
             service.getCustomerLookup_MULIPLEROWPort(new java.net.URL(END_POINT_URL);
```

The variable END_POINT_URL contains the URL of the WSDL.

Step 3. Create the Request Object

You must create a request object and any required parameter to be passed to the web service. In the sample client application, the following code creates a lookup request object:

```
CustomerLookupRequest request = new CustomerLookupRequest();
request.setCustomerID_in(CustomerID);
```

Step 4. Send the Request and Handle the Response

After you create the request object, pass it to the port operation. The web service sends back a response. You can handle the response based on your requirements.

In the sample client application, the following code passes the request object to the port and displays the response:

```
CustomerLookupResponse[] response =
port.customerLookup_MULTIPLEROWOperation(requestOperation);
System.out.println();
if (response[0].getCustomerID_out() == 0)
   {
    System.out.println("Customer(s) with the ID as " + CustomerID + " does not
exist!!!");
   }
else
   {
    System.out.println("***** Customer(s) that matches with the Customer ID is/are ...");
    for (int i = 0; i < response.length; i++)
       {
        System.out.println("***** Customer ID: " + response[i].getCustomerID_out());
        System.out.println("***** Customer Name: " + response[i].getCustomerName_out());
        System.out.println("***** Customer Age: " + response[i].getCustomerAge_out());
        System.out.println("***** Customer Gender: " +
                           response[i].getCustomerGender_out());
        System.out.println("***** Customer Address: " +
                           response[i].getCustomerAddress_out());
        if (i < response.length - 1) System.out.println ();
       }
   }
```

Using Parameter Arrays

In CDI-PC, a parameter represents a value you can change between sessions, such as a database connection or a source or target file. You can create parameters associated with a workflow or session to provide flexibility each time you run a workflow or session.

For a web service client application, you can define the values for parameters associated with a workflow or session in a parameter file or a parameter array. To use the parameters in a parameter file, specify the parameter file name in the client application. The parameter file must be accessible to the Integration Service. To use a parameter array, provide the parameter values in the elements of the parameter array in the client application.

For example, a request to start a workflow or task can specify the parameters associated with the workflow or task with the name of a parameter file or the list of parameters and values in parameter array.

Parameter Array Definition

The parameter definition in a SOAP request consists of the scope, name, and value of the parameter. When the Integration Service runs the workflow or task, it uses the parameters in an array the same way it uses parameters in a parameter file.

The WSDL contains the following definition for the parameter array elements:

```
<complexType name="Parameter">
   <sequence>
      <element name="Scope" type="xsd:string" />
      <element name="Name" type="xsd:string" />
      <element name="Value" type="xsd:string" />
   </sequence>
</complexType>

<complexType name="ParameterArray">
   <sequence>
      <element maxOccurs="unbounded" minOccurs="0" name="Parameters"
               nillable="true" type="impl:Parameter" />
   </sequence>
</complexType>
```

For example, a parameter file has the following parameters:

```
[s_m_A]
$a=1
$b=2
$c=3
[WSH_Folder.s_m_B]
$d=4
```

The SOAP request for a web service call to the StartWorkflow operation with the same parameters in a parameter array would include the following elements:

```
<StartWorkflow>
…
   <Parameters>
      <Parameter>
         <Scope>s_m_A</Scope>
         <Name>$a</Name>
         <Value>1</Value>
      </Parameter>
      <Parameter>
         <Scope>s_m_A</Scope>
         <Name>$b</Name>
         <Value>2</Value>
      </Parameter>
      <Parameter>
         <Scope>s_m_A</Scope>
```

```
            <Name>$c</Name>
            <Value>3</Value>
         </Parameter>
         <Parameter>
            <Scope>WSH_Folder.s_m_B</Scope>
            <Name>$d</Name>
            <Value>4</Value>
         </Parameter>
…
</StartWorkflow>
```

The WorkflowRequest and TaskRequest types contain ParameterArray elements. You can specify any number of parameters in a parameter array.

The following sample code from a web service client application in Axis shows how to create the parameter array in a WorkflowRequest:

```
Parameter[] parameters = new Parameter[4];

Parameter param1 = new Parameter();
Param1.setScope("s_m_A");
Param1.setName($a");
Param1.setValue("1");
Parameters[0] = param1;

Parameter param2 = new Parameter();
Param2.setScope("s_m_A");
Param2.setName("$b");
Param2.setValue("2");
Parameters[1] = param2;

Parameter param3 = new Parameter();
Param3.setScope("s_m_A");
Param3.setName("$c");
Param3.setValue("3");
Parameters[2] = param3;

Parameter param4 = new Parameter();
Param4.setScope("WSH_Folder.s_m_B");
Param4.setName("$d");
Param4.setValue("4");
Parameters[3] = param4;

WorkflowRequest wfReq = new WorkflowRequest();
wfReq.setParameters(parameters);
```

You can use parameter arrays in the following operations:

- startWorkflow
- startWorkflowFromTask
- recoverWorkflow
- startTask

Rules and Guidelines for Using Parameter Arrays

Use the following rules and guidelines when you use a parameter array in a web service request:

- **Use a parameter file OR a parameter array.** Do not specify a parameter file name and a parameter array in the SOAP request when you make a web service operation call. If you specify both a parameter file and parameter array in the SOAP request, the Web Services Hub returns a fault message warning that the request specifies a parameter list and a parameter file.

- **When a parameter file and a parameter array are defined, the Integration Service uses the value of the parameter from the parameter array in a SOAP request.** The Integration Service uses the value of the parameter defined in the parameter array when the following conditions are true:
  - You specify a parameter array in a web service request to start a workflow.
  - The workflow has an associated parameter file defined in the workflow properties.

Adding Security to a Client Request

The Web Services Hub uses the following types of security for web services:

- **User credential.** To get credentials for a request, the client must log in to the CDI-PC repository that contains the web service to run. The login generates a session ID that the web service client must include in the SOAP request.
- **User name token.** Web service security that uses a user name token and is based on the OASIS web service security standards, which includes a set of SOAP extensions to ensure content integrity and security for SOAP messages.

  User name token is the default security option for protected web services. By default, WSDLs generated by the Web Services Hub for protected web services contain a security header with the UsernameToken element.

  For more information about the OASIS web service security standards, read the web services security specifications on the OASIS web site:

  ```
  http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-sx
  ```

Note: If a client application sends a login request to run a web service, send an explicit logout request after the response is received. Login requests without corresponding logout requests can cause a memory leak in the Repository Service and Web Services Hub processes.

UsernameToken in the SOAP Request

When you build a client application based on the WSDL generated by the Web Services Hub, the request object contains the UsernameToken element in the header by default.

The UsernameToken element in the SOAP request can have one of the following password security:

- **Plain text password.** Includes a password in plain text.
- **Hashed password.** Includes an encrypted password hashed using the MD5 or SHA-1 hash function.
- **Digested password.** Includes an encrypted password that is hashed with a nonce value and a timestamp.

Include the user password in the Password element of the UsernameToken.

The Password element has a Type attribute to indicate the type of password security used. If the Type attribute is omitted, the password type defaults to *PasswordText*.

Note: If the Informatica domain uses Kerberos network authentication, you cannot use hashed or digested passwords in the SOAP request.

Plain Text Password

The UsernameToken element includes the following child elements:

- **Username element.** Contains a user name that can be found in the CDI-PC Native security domain or any LDAP security domain. The default security domain is the Native security domain. If the user name belongs to the Native security domain, the Username element does not require the name of the security domain. If the user name belongs to an LDAP security domain, the user name must be preceded by the name of the security domain and a slash (/).

  The following table of Username element examples shows the format to use to indicate the security domain of the user account:

Value of the Username Element	Security Domain
<UsernameToken> **<Username>Native/Administrator</Username>** <Password>Administrator</Password> </UsernameToken>	Native
<UsernameToken> **<Username>/Administrator</Username>** <Password>Administrator</Password> </UsernameToken>	Native
<UsernameToken> **<Username>Administrator</Username>** <Password>Administrator</Password> </UsernameToken>	Native
<UsernameToken> **<Username>LDAPAdm/Administrator</Username>** <Password>Administrator</Password> </UsernameToken>	LDAP security domain named *LDAPAdm*.

- **Password element.** Contains the password in plain text. The Type attribute of the Password element can be omitted or set to *PasswordText*.

Hashed Password

The UsernameToken element includes the following child elements:

- **Username element.** Contains a user name that can be found in the CDI-PC Native security domain.
- **Password element.** Contains a hashed password. The password must be hashed with the MD5 or SHA-1 hash function and encoded to Base64. The Type attribute of the Password element can be omitted or set to “PasswordText”.

  The following code shows an example of the security header for a request that uses a hashed password:

```
<soapenv:Header>
    <!--  UsernameTokens -->
    <inf:Security>
        <UsernameToken>
            <Username>Native/Administrator</Username>
            <Password>Ntm58Cxf7SBOQAz3OlsTq1nv-D7</Password>
        </UsernameToken>
```

```
        </inf:Security>
    </soapenv:Header>
```

Using Third-Party Tools to Create a Hashed Password

You can use third-party tools, such as OpenSSL and the Java MessageDigest class, to create a hashed password.

OpenSSL on UNIX

To use OpenSSL to create a hashed password on a UNIX machine, run OpenSSL with the message digest command dgst.

The following example shows how to create a hashed password for the password string Administrator using the MD5 hash function and encoded in Base64:

```
echo -n "Administrator" | openssl dgst -md5 -binary | openssl base64
```

The following example shows how to create a hashed password for the password string Administrator using the SHA-1 hash function and encoded in Base64:

```
echo -n "Administrator" | openssl dgst -sha1 -binary | openssl base64
```

Note: The echo command in these examples adds a newline character to the string. The -n option in the commands removes the newline character.

OpenSSL on Windows

To use OpenSSL to create a hashed password on a Windows machine, run OpenSSL with the message digest command dgst.

The following example shows how to create a hashed password using the SHA-1 hash function and encoded in Base64:

```
openssl dgst -sha1 -binary -out <output file name> <input file name>
openssl enc -base64 -in <output file name>
```

The input file contains the password string that you want to hash. OpenSSL writes the hashed password to the output file.

Java MessageDigest

The following example shows how to use the Java MessageDigest class to create a hashed password using the MD5 hash function and encoded in Base64:

```
public static String md5hash(String password) throws Exception{
   MessageDigest digest = java.security.MessageDigest.getInstance("MD5");
   digest.reset();
   digest.update(password.getBytes());
   byte[] hash = digest.digest();
   return new String(org.apache.commons.codec.binary.Base64.encodeBase64(hash));
}
```

The following example shows how to use the Java MessageDigest class to create a hashed password using the SHA-1 hash function and encoded in Base64:

```
public static String sha1hash(String password) throws Exception{
   MessageDigest digest = java.security.MessageDigest.getInstance("SHA-1");
   digest.reset();
   digest.update(password.getBytes());
   byte[] hash = digest.digest();
   return new String(org.apache.commons.codec.binary.Base64.encodeBase64(hash));
}
```

Digested Password

The UsernameToken element includes the following child elements:

- **Username element.** Contains a user name that can be found in the CDI-PC Native security domain.
- **Password element.** Contains a digested password. The password is the value generated from hashing the password concatenated with the nonce value of the Nonce element and the timestamp in the Created element. The password must be hashed with the SHA-1 hash function and encoded to Base64.

  For digested password security, the Type attribute of the Password element must be set to *PasswordDigest*.
- **Nonce element.** Contains a nonce value, which is a random value that can be used only once.
- **Created element.** Contains a timestamp value that indicates the time when the request was created. The timestamp uses the UTC format, `yyyy-MM-dd'T'HH:mm:ss.SSS'Z'`. For example: `2008-08-11T18:06:32.425Z`.

The nonce value you include in a SOAP request can be used only once. By default, it is valid for 300 seconds (five minutes) after the time that the request is created, as indicated by the value in the Created element. The client application must send the request within the time that the nonce value is valid. For example, the Created value indicates that the request was created at 10:00 a.m. The request is valid from 10:00 a.m. to 10:05 a.m. If the client application sends the request to the Web Services Hub before 10:00 a.m. or after 10:05 a.m., then the request and the nonce value are not valid and the request will fail.

The digested password uses the standard OASIS password digest algorithm:

```
Password_Digest = Base64 ( SHA-1 ( nonce + created + password ) )
```

You can use any tool to generate the nonce value, timestamp, and the digested password.

The following code shows an example of the security header for a request that uses a digested password:

```
<soapenv:Header>
    <!--  UsernameTokens -->
    <inf:Security>
        <UsernameToken>
            <Username>Administrator</Username>
            <Password Type="PasswordDigest"> Xty5lCAf5SVO0AY3OtsYq7nv/DI=</Password>
            <Nonce>KjsaeiuDFKJEwkr4332rL=</Nonce>
            <Created>2008-08-12T01:11:47.013Z</Created>
        </UsernameToken>
    </inf:Security>
</soapenv:Header>
```

Using Third-Party Tools to Create a Digested Password

You can use a third-party tool such as the Java MessageDigest class to create a digested password.

The following example shows how to use the Java MessageDigest class to create a digested password with a timestamp and nonce value and encoded in Base64 :

```
public static String oasisDigest(String password, String nonce) throws Exception{
   SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS'Z'");
   String created = sdf.format(new Date());
   System.out.println("Created : " + created);
   System.out.println("Nonce : " + new
String(org.apache.commons.codec.binary.Base64.encodeBase64(nonce.getBytes())));
   String toDigest = nonce + created + password;
   MessageDigest digest = java.security.MessageDigest.getInstance("SHA-1");
   digest.reset();
   digest.update(toDigest.getBytes());
   byte[] hash = digest.digest();
   return new String(org.apache.commons.codec.binary.Base64.encodeBase64(hash));
}
```

If you use a web service testing tool such as soapUI to test client applications, you can use the tool to generate the digested password for the client request.

CHAPTER 5

Working with Web Service Sources and Targets

This chapter includes the following topics:

- Web Service Sources and Targets Overview, 35
- Understanding Web Service Sources and Targets, 36
- Importing a Web Service Source or Target Definition, 38
- Creating a Source or Target Definition, 41

Web Service Sources and Targets Overview

When you create a mapping to use in a web service workflow, the source and target for the mapping must define the web service input and output messages. The web service source defines the input message of a web service operation and represents the metadata for a web service SOAP request. The web service target defines the output message of a web service operation and represents the metadata for a web service SOAP response.

The Web Services Description Language (WSDL) describes the input and output message of the web service. If you have a WSDL for the web service workflow you want to create, you can import the source and target definitions from the WSDL. If you do not have a WSDL, you can create the input and output messages based on columns in a relational or flat file source or target. Or you can create the input and output messages from columns that you define.

Use the Designer to create the web service source and target definitions. You can create the web service source and target definitions in the following ways:

- **Import definitions from a WSDL.** Import the source or target definition for an operation defined in a WSDL. When you import a source definition, the Designer imports the definition of the input message. When you import a target definition, the Designer imports the definition of the output and fault messages.
- **Create definitions from relational or flat file sources and targets.** You can create the web service source and target definitions based on relational or flat file source and target definitions defined in the folder. You can also manually define the columns and specify the datatype and column size. You do not need a WSDL when you create the web service source and target definitions from relational or flat file sources and targets or when you manually define columns.

Understanding Web Service Sources and Targets

Similar to XML sources and targets, web service source and target definitions are organized into XML views. XML views are groups of columns that represent the elements and attributes defined in the input and output messages.

When you import source and target definitions, the Designer generates XML views based on the elements in the input or output messages defined in the WSDL. It also generates views for mime attachments to the input or output messages.

When you create web service source and target definitions, the Designer creates XML views based on the columns defined in the relational or flat file sources or targets or the columns that you manually define.

XML Views and Groups

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

- **Envelope.** XML view that corresponds to the SOAP envelope and body elements. The Envelope view is the main view that contains a primary key and the ports for the input or output message.

  If the body message parts are simple, the Designer generates an envelope view.

  If the body message parts are complex, the Designer can generate additional body views:

  - **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.
  - **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 imported in entity relationship mode.
- **Header.** XML view that corresponds to a SOAP header element. If the header message parts are complex, the Designer can split the header view into separate element and type views.
- **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 imported in normalized hierarchical relationship mode or entity relationship mode.
- **Attachment.** Attachment view generated for a WSDL that contains a mime attachment. The attachment view has an *n*:1 relationship with the envelope view.

  The Designer generates an attachment view for a web service source or target definition based on a WSDL that contains an element definition for a mime attachment.

Related Topics:

- "WSDL Attachments" on page 56

Source Definition

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

The following table shows the XML views the Designer can generate for a web service source definition:

Import Mode	Envelope	Element	Type	Fault	Attachment
Entity Relationship	yes	yes	yes	no	yes
Normalized Hierarchical Relationship	yes	yes	no	no	yes

Target Definition

When you create a target definition for an operation in a WSDL, the Designer imports the output message and any fault message associated with the operation. If the function within an operation results in different faults, the Designer creates multiple fault views in the target definition. A fault message represents an error processing the request.

The following table shows the XML views the Designer can generate for a web service target definition:

Import Mode	Envelope	Element	Type	Fault	Attachment
Entity Relationship	yes	yes	yes	yes	yes
Normalized Hierarchical Relationship	yes	yes	no	yes	yes

Note: To create separate target definitions for fault messages, configure the web services option in the CDI-PC Designer.

Rules and Guidelines for Importing or Creating Web Service Sources and Targets

Use the following rules and guidelines when you import or create web service sources and targets:

- **Use a WSDL for elements with complex relationships.** To create a web service source or target with a complex element relationship, first create a WSDL to define the element hierarchy, and then import the source or target from the WSDL. Use a WSDL to create a web service source or target that contain multiple occurrences of elements or that contain elements of complex type.
- **Manually define simple web service source or target definitions.** To create a web service source or target with a simple set of columns and without nested elements, create the definition manually or use a relational or flat file source or target definition. You can specify that all columns in the web service source or target definition occur multiple times.
- **Create the source and target definitions in the same process.**Create a web service target definition from a relational or flat file source or target at the same time you create the source definition. To create the source and target of a web service mapping at the same time, verify that the Create Source and Create Target options are selected on Create Web Service Definition window. For example, in the Source Analyzer, select Sources > Web Service Provider > Create Web Service Definition. On the Create Web Service Definition window, select the Create Target option.
- **Use the same method to create the source and target definitions for a request-response mapping.** If you create a request-response web service mapping, create the source and target definitions using the same method. For example, if you import the source definition from a WSDL, import the target definition from the same operation in the WSDL. If you create the source definition by defining the columns or by using relational or flat file sources and targets, create the target definition using the same method.

- **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. You cannot define fault views in a target definition if you create it based on a flat file or relational source or target. If you define a web service target definition based on a flat file or relational source or target, the Web Services Hub can generate fault messages for system faults only.
- **The input and output message in the WSDL must have the same encoding style.** If you import web service sources and targets 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.

  If you create the web service source or target definition manually or based on relational or flat file sources or targets, the Designer uses the Document/Literal encoding style for the input and output messages.
- **Elements in the WSDL cannot refer to a standard W3C XML schema.** You cannot import web service source and target definitions from a WSDL that contains an element that refers to a standard W3C XML schema.
- **Import a WSDL with empty complexType elements in entity relationship mode.** If the WSDL contains complexType elements that will not contain values in the input message of the web service request, import the source and target definitions from the WSDL in entity relationship mode. If the source and target definitions are imported from the WSDL in normalized hierarchical mode, the web service generates a fault response if you send a request with an empty complexType element.
- **Import sources and targets from a WSDL with valid XML syntax.** If you import from an invalid WSDL, the Designer cannot correctly display the WSDL definition in the Web Services Wizard. In some cases, the Designer does not generate error messages but partially parses the WSDL and displays only the services and operations that were successfully parsed. If you import from a WSDL and the Web Services Wizard does not display the correct WSDL definition, open the WSDL as an XML file and verify that the syntax is correct.
- **Define a two dimensional array with the correct syntax.** If you define a complexType element in the WSDL as a two dimensional array of string, use the following syntax:

  ```
  wsdl:arrayType="xsd:string[][]"
  ```

  You cannot import web service source and target definitions from a WSDL that contains a two dimensional array defined using a different syntax.
- **You cannot import web service sources and targets from a WSDL that generates a large number of XML views.** The limit for the number of XML views that can be generated from a WSDL file is 400. To create web service sources or targets with more than 400 XML views, create the groups manually in the WSDL workspace.

Related Topics:

- "SOAP Fault Handling" on page 15

Importing a Web Service Source or Target Definition

If you have a WSDL that defines the input and output messages of a web service you want to create, you can import the source and target definitions from the WSDL. You can use the Web Services wizard to import the web service source or target definition from the WSDL. You can import the source and target definitions in the same process.

Import Modes

You can create XML views for a web service source or target definition based on the mode in which you import the WSDL:

- **Entity relationship.** This is the default import mode for source or target definitions imported from a WSDL. Use this import mode 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.
- **Normalized hierarchical relationship.** 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.
- **Do not generate XML views.** Use this import mode to create the source or target definition without defining the XML views. You can use the WSDL workspace to add the XML views and ports.

Related Topics:

- "Importing from a WSDL Without Creating XML Views" on page 40

Message ID

To use web service source and target definitions in a staged mapping, you must include a message ID in the web source and target definitions. The Web Services Hub uses a message ID as a primary key to bind the requests and responses for a web service. For example, the first session reads from a web service source and writing to a relational target. The second session uses the relational target as a source and writes to a web service target. The Web Services Hub uses the message ID to associate the input message of the request in the first session to the output message in the response of the second session.

You must also include a message ID in the web source and target definitions when you run a web service session on a grid. When you run a web service session on a grid, the Integration Service distributes session threads to multiple DTM processes on nodes in the grid. The Integration Service uses the message ID to associate the web service input and output messages across the nodes.

If you add the message ports when you create a web service source or target definition, the Designer adds message ID and client ports to the envelope view.

The following table describes the message ID and client ports added to the envelope view:

Port Name	Description
MessageID	Web Services Hub generates the message ID when it receives a request. It uses this ID to correlate the incoming request with the outgoing response.
ClientIP	TCP/IP address of the web service client.

Advanced Options

When you import a web service source or target from a WSDL, you can specify the length of fields with undefined length and the naming convention for the XML columns.

The following table shows the advanced options you can set when you import a web service source or target from a WSDL:

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. CDI-PC uses the following format for the name of the XML column: *NameOfElement_NameOfAttribute* - Prefix the XML view name for every XML column. CDI-PC uses the following format for the name of the XML column: *NameOfView_NameOfElement* - Prefix the XML view name for every foreign-key column. CDI-PC 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. CDI-PC truncates column names longer than 80 characters. If a column name is not unique, CDI-PC 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.

Importing from a WSDL Without Creating XML Views

When you import a source or target definition from a WSDL and you want to manually define the XML views and ports, you can create an empty source or target definition.

For example, you 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. The target definition is not affected. You can import the target definition and create the XML views.

To import a source or target definition from a WSDL without creating XML views, select Do Not Generate XML views 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 ports, and the relationship between views. Right-click the title of the source or target definition and select WSDL Workspace.

Related Topics:

- "Editing Definitions in the WSDL Workspace" on page 46

Importing a Web Service Source or Target Definition from a WSDL

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 with RPC/SOAP Encoded or Document/Literal encoding style.

Note: When the Designer imports a web service target definition, it names the definition based on the operation and the target type, such as output or target. If you rename the definition after the import, you can verify the target type on the Metadata Extensions tab.

To import a web service source or target definition from a WSDL:

1. To import a source definition, in the Source Analyzer, click Sources > Web Service Provider > Import From WSDL. To import a target definition, in the Target Designer, click Targets > Web Service Provider > Import From WSDL.
2. Click Advanced Options.

   The XML Views Creation and Naming Options window appears.
3. Specify the default length for fields with undefined lengths and select how to generate the names of the XML columns.
4. Choose to import from a local file or a URL.

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

   If you import from a local file, select a WSDL file in a local folder and click Open.
5. Select the operation defined in the WSDL for which you want to create the source or target definition.

   Note: If you import from a WSDL that contains errors, the Web Services Wizard (Step 1) window cannot correctly display the list of services, bindings, ports, or operations defined in the WSDL. The window displays an empty or partial WSDL definition tree. For example, if the WSDL contains an error in a type definition, the window displays an empty WSDL definition tree.
6. Click Next.

   The Web Services Definition Creation Options dialog box appears.
7. Select the import mode.

   The import mode determines the type of XML views to generate. You can generate XML views as entity relationships or as hierarchical relationships. The default import mode is entity relationship.
8. To create a source or target definition without any views or ports defined, select Do not generate XML views.

   If you do not generate the XML views, the Designer creates an empty source or target definition. The source or target definition contains no views or ports. You must use the WSDL workspace to manually add the views and ports for the source or target definition.
9. Select whether to add message and header ports to the source or target definition.

   For more information, see "Message ID" on page 39.
10. To generate source and target definitions in the same import process, select both the Create Source and Create Target options.

    The Designer creates the source and target definitions based on the selected options.
11. Click Finish.

    The web service source or target definition appears in the workspace.

Creating a Source or Target Definition

If you do not have a WSDL from which to import a web service source or target definition, you can create the definition from a relational or flat file source or target. You can also manually define the ports for the source or target and specify the datatype and the occurrence.

When you create web service source and target definitions from other sources and targets or from manually defined columns, the Designer creates the views in the entity relationship mode. You cannot create web service sources or targets from other sources or targets in hierarchical relationship mode.

You can create web service source and target definitions from relational sources and targets created from the following relational databases:

- Oracle
- DB2
- Informix
- Teradata
- Microsoft SQL Server
- Sybase

When you create a web service source or target definition from relational or flat file sources or targets, the Designer lists the sources and targets available in the folder, including shortcuts to sources and targets.

Create the source and target definitions in the same process. After you create a web service source or target based on relational or flat file source or target, you can edit the columns of the source or target definition in the Designer workspace.

When you import a web service source or target definition from a WSDL, you can select the following options:

- Multiple-occurring Elements
- Message Ports

Related Topics:

- "Editing Definitions in the Designer Workspace" on page 45

Multiple Occurring Elements

When you create web service source and target definitions from a relational or flat file source or target, you must indicate whether the columns in the source or target definitions occur multiple times. Selecting the Multiple Occurring Elements option indicates that the columns as a group occur multiple times. The columns in the list represent an array.

When you edit the web service source or target created with this option, you cannot change the multiple occurring property of the columns of the source or target definition.

Message Ports

You can include a message ID in the web source and target definitions. The Web Services Hub uses a message ID as a primary key to bind the requests and responses for a web service.

Note: When you edit the web service source or target definition in the Designer workspace, the message and client ports do not appear in the Web Service Definition tab. You cannot modify the message and client ports.

Related Topics:

- "Message ID" on page 39
- "Web Service Definition Tab" on page 46

Creating a Source or Target from a Relational or Flat File Source or Target

Use this procedure to create a web service source or target in one of the following ways:

- From a flat file source or target
- From a relational source or target
- By manually adding a column and specifying its name, datatype, and precision

To create a web service source or target from a relational or flat file source or target:

1. To create a web service source, in the Source Analyzer, click Sources > Web Services Provider > Create Web Service Definition. To create a web service target, in the Target Designer, click Targets > Web Services Provider > Create Web Service Definition.
2. Enter a name for the web service mapping where you plan to use the source and target definition.

   The Designer uses the web service definition name as the name of the source and target definitions. It adds the suffix *_input* to the source definition name or *_output* to the target definition name.

   If you know the columns you want to include in the web service source or target definition, you can add them directly to the source or target definition. If you have relational or flat file sources or targets in the folder, you can create the source or target definition based on the relational or flat file columns.
3. To add a column to the list, click the Add button and specify the column name, datatype, and precision.

   The Web Services Hub ignores the Not Null property when you create a web service source with multiple occurring elements or when you create a web service target.
4. To create a web service source or target based on the columns in a relational or flat file source or target definition, click the Import from Source/Target button and select a source or target definition.

   The Designer lists the columns found in the selected source or target.
5. Click OK.
6. Edit the column names and the precision of string columns if necessary.

   You can add columns to the definition and define their data types and properties. You can delete columns you do not want to use.

   You can specify whether the columns you define in the source or target definition occur once or multiple times.
7. If the columns occur multiple times, select Multiple occurring elements.

   This option indicates that the columns as a group occur more than once. When you select this option, the Designer generates an element view that contains all the columns.
8. If you are creating a source definition and you also want to create the target definition, click Create Target and repeat steps 3 to 7 to add the ports to the target definition.

   If you are creating a target definition and you also want to create the source definition, click Create Source and repeat steps 3 to 7 to add the ports to the source definition.
9. To add message and client ports to the source or target definition, click Add message ports.

   The Designer adds the message and client ports to the envelope view of the source or target definition. If you are creating source and target definitions in the same process, the Designer adds the message and client ports to the envelope views of the source and target definitions.
10. Click OK.

    The Designer creates the web service source or target definition.

Review the XML views in the source and target definitions to verify that the views and ports match your web service mapping requirements. To add, delete, or modify the columns, edit the source or target definition in the Designer workspace.

Related Topics:

- "Multiple Occurring Elements" on page 42
- "Message ID" on page 39
- "Editing Definitions in the Designer Workspace" on page 45

CHAPTER 6

Editing Web Service Sources and Targets

This chapter includes the following topics:

- Editing Web Service Sources and Targets Overview, 45
- Editing Definitions in the Designer Workspace, 45
- Editing Definitions in the WSDL Workspace, 46

Editing Web Service Sources and Targets Overview

You can edit the web service source or target definition based on how you create it:

- **Source or target definition imported from a WSDL.** If you import the web service source or target definition from a WSDL, you can edit the source or target definition in the WSDL workspace. You can use the WSDL workspace to add, modify, or delete views in the source or target definition.

  You can view the definition and edit some properties in the Designer workspace.
- **Source or target definition created from a relational or flat file source and target.** If you create a web service source or target definition based on a relational or flat file target, you can edit the columns in the Designer workspace.

  You can view the source or target definition in the WSDL workspace. You cannot edit the source or target definition in the WSDL workspace.

Editing Definitions in the Designer Workspace

In the Designer workspace, you can add descriptions or specify links to business documentation for a web service source or target definition at any time. If you create a source or target definition manually or based on a relational or flat file target, you can modify the list of columns in the source or target definition. When you make changes to the columns, the changes are immediately reflected in the XML views.

To view or edit the properties of a source or target definition, double-click the source definition in the Source Analyzer or the target definition in the Target Designer. Or you can right-click the title of the source or target definition and select Edit.

You can view or edit the web service source and target definition in the following tabs:

- **Table.** On the Table tab, you can provide the owner name and description, and you can change the name of the definition. You cannot change the table type.
- **Columns.** On the Columns tab, you can edit the precision for String datatypes. You can also add business names and column descriptions.
- **Attributes.** On the Attributes tab, you can view attribute values for each column in a source or target definition.
- **Web Service Definition.** This tab appears if you edit a source or target definition created from a relational or flat file target. You can add, edit, or delete a column in the source or target definition. The changes you make immediately appear in the Columns tab.

Table Tab

The Table tab displays the table information for the source or target definition. You can change the name of the source or target definition. You can modify the owner and description of the source or target definition.

Columns Tab

The Columns tab displays the XML views in the web service source or target definition. You can edit the precision for String and Binary datatypes, and you can add business names and column descriptions.

The default precision for the String datatype is the value at which the precision of infinite length data is set during the WSDL import process. You can set the precision for the String datatype when you import a source or target definition from a WSDL. You can set the precision for individual columns when you edit the definition.

Note: The Mapping Designer invalidates mappings that use source and target web service definitions with a total column length greater than 500 MB.

Attributes Tab

The Attributes tab is a read-only tab that displays the XPath and XMLDataType values for each field in the web service source or target definition. If the definition has an Attachment group, the Attributes tab displays the MIME type in the data field.

Web Service Definition Tab

The Web Service Definition tab appears for web service source or target definitions that are manually defined or are based on relational or flat file sources or targets. You can add or delete columns in the source or target definition. You can change the names and datatypes of the columns and modify the precision and scale for specific datatypes. You can also specify whether the columns occurs once or multiple times.

When you make changes to the columns in the Web Service Definition tab, the changes are reflected in the Columns tab.

Editing Definitions in the WSDL Workspace

If you import a source or target definition from a WSDL and create XML views, 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 generate 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 WSDL Workspace.

The WSDL workspace is equivalent to the XML Editor. You use the WSDL workspace the same way you use the XML Editor. However, the WSDL workspace performs validation on changes to the views specific to web service source and target definitions. Also, you cannot perform some tasks in the WSDL workspace that are allowed in the XML workspace.

You cannot perform the following tasks in the WSDL workspace:

- Pivot columns when you add or edit an XML view in a web service source or target definition.
- Create XPath query predicates to filter elements or attributes in an XML view.
- Preview XML data.
- Add a FileName column to an XML view.
- Add a reference port.
- Recreate entity relationships.
- Set XML view options in the Columns window.

Rules and Guidelines for the WSDL Workspace

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

- 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 an input message. It cannot define views for an output or fault message.
- The name for the root group and the primary key of the root group for a source or target definition must use the following naming convention, where <*NameString*> can be any alphanumeric string:
  - The root group must be named Message or X_<*NameString*>_Envelope.
  - The primary key for the root group must be named PK_Message or PK_<*NameString*>_Envelope.
  - The <*NameString*> for the root group and its primary key must be the same.
- A target definition must define views for an output or fault message. It cannot define views for an input message.
- You can define elements with the type anyType or any. You can create a string port for an element of type anyType or map it to an element of type complexType.
- The envelope view in a target definition must contain a view root and a view row as the envelope node.
- 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 cannot use any of the following strings as a namespace prefix:
  - mime
  - wsdl

- soap
- soapenc
- http

CHAPTER 7

Working with Web Service Mappings

This chapter includes the following topics:

- Working With Web Service Mappings Overview, 49
- Types of Web Service Mappings, 50
- Generating a Mapping from a WSDL, 51
- Generating a Mapping from a Relational or Flat File Source or Target, 52
- Generating a Mapping from a Transformation or Mapplet, 53
- Editing a Target Instance in a Web Service Mapping, 54
- Attachments, 55

Working With Web Service Mappings Overview

After you create the web service source and target definitions, create the mapping to determine how the Integration Service handles the data received in the web service request and send out the web service response. A web service mapping receives the input message as a SOAP request, transforms the data, and sends the output message as a SOAP response.

You can create a web service mapping in the Mapping Designer in the same way you create other CDI-PC mappings. Add the web service source and target definitions and transformations to the mapping.

You can also generate a mapping that includes the web service source definition, Source Qualifier transformation, and web service target definition. The CDI-PC Designer provides several ways to generate a web service mapping.

You can generate a web service mapping in the following ways:

- **By importing source and target definitions from a WSDL.** You can create a mapping from a WSDL in the same way you create a web service source or target from a WSDL.
- **From a relational or flat file source or target definition.** You can create a mapping from a relational or flat file source or target in the same way you create a web service source or target from a relational or flat file source or target.
- **From a transformation or mapplet.** You can create a mapping from a reusable transformation or a mapplet with a single input and a single output.

After you generate the mapping, you can add more transformations, links, and any other mapping objects you require to complete the web service mapping.

Types of Web Service Mappings

You can create a mapping to receive a message from a web service client, transform the data, and send the response back to the web service client or write it to any target CDI-PC supports. Based on the source and target definitions, the Integration Service can receive and send an attachment as part of the SOAP request.

You can also create a mapping with flat file or XML sources and targets and use it in a web service workflow. This allows you to receive message data through a SOAP call by attachment instead of by reading it from a file.

The mapping you create depends on the type of web service that you want to run:

- **Request-response web service.** A request-response web service receives an incoming request from the web service client, transforms the data, and sends the response back to the web service client. A request-response web service uses both a web service source and a web service target.

  You can create one mapping or multiple mappings to process a request-response web service:

  - **One mapping.** Create one mapping that contains both the web service source and web service target definitions. The Integration Service receives an incoming request, transform the data, and send the response back in a single session.
  - **Multiple mappings.** Create multiple mappings to stage data before sending a response back to the web service client. You can create a workflow that contains a session for each mapping.
- **One-way web service.** If you receive updates and notifications from a web service client, but do not need to send back a response, you can create a one-way mapping. A one-way mapping uses a web service client for the source. The Integration Service loads data to a target, often triggered by a real-time event through a web service request.

The web service source and target definitions you include in the mapping depend on the type of mapping you create.

The following table describes the web service source and target definitions you use based on the mapping type:

Mapping Type	Web Service Source	Web Service Target
Request-Response	Must have one instance of a web service source definition.	Must have one instance of a web service target definition. Can have multiple fault views in the target definition.
One-way	Must have one instance of a web service source definition.	Contains no web service target definition.

Related Topics:

- "Attachments" on page 55

Request-Response Mappings

A request-response mapping uses a web service source and a web service target.

If you create a request-response mapping, use source and target definitions created in the same method. If you import the source definition from a WSDL, import the target definition from the same operation in the WSDL. If you create the source definition by defining the columns or by using relational or flat file sources and targets, create the target definition using the same method.

To ensure that the web service source and target definitions are created using the same method, create the source and target definitions in one process.

Note: If you do not import source and target definitions from the same operation in the WSDL or if you do not create them using the same method, you can get unexpected results.

You can use an SQL transformation to update a database or to retrieve multiple database rows in a request-response mapping. The SQL transformation can return multiple database rows to the target. When database errors occur in processing, the SQL transformation receives the errors from the database and sends the error text to the target.

For an example of a web service that uses an SQL transformation to get multiple rows, see the real-time web services examples shipped with CDI-PC. By default, the real-time web services sample programs are installed in the following directory:

```
/<CDI-PCInstallDir>/server/samples/WebServices/samples/RealTimeWebServices
```

Related Topics:

- "Creating a Source or Target Definition" on page 41

Staged Mappings

If you want to run a request-response session, but you need to stage the data first, you can create multiple mappings to process the data.

For example, you receive message data that you need to process. You must make an asynchronous call to an external system through WebSphere MQ. You create the following mappings:

1. Create a request mapping with a web service source definition. This mapping writes to a flat file target and a WebSphere MQ target. You write all message data to both targets.

   An external application receives messages from the WebSphere MQ target, processes them, and sends messages to another WebSphere MQ queue.
2. Create a response mapping with a web service target definition. This mapping uses the flat file target in the first mapping as a source. It also uses the WebSphere MQ queue with the processed data as a source.

The Web Services Hub uses a message ID to connect the requests and responses in a staged mappings. To use web service source and target definitions in a staged mapping, you must include a message ID in the web source and target definitions.

Related Topics:

- "Message ID" on page 39

Generating a Mapping from a WSDL

You can generate a web service mapping by importing the web service source or target from a WSDL that you can access locally or through a URL.

When you generate a web service mapping by importing source and target definitions from a WSDL, the Designer creates the source definition from the input message of the operation you select. It creates the target definition from the output message of the operation you select.

The web service mapping generated from a WSDL contains the following objects:

- Web service source definition
- Source qualifier
- Web service target definition

The Designer links the ports from the source instance through the target instance. To complete the mapping, add the transformations and other mapping component necessary for the web service you want to create.

To generate a mapping from a WSDL:

1. In the CDI-PC Designer, open the Mapping Designer.
2. Click Mappings > Create Web Service Mappings > Import from WSDL.

   The procedure to generate a web service mapping by importing sources and targets from a WSDL is the same as the procedure to create web service sources or target definitions from a WSDL. For more information, see "Importing a Web Service Source or Target Definition" on page 38.
3. Save the mapping to the repository.

Generating a Mapping from a Relational or Flat File Source or Target

You can generate a web service mapping based on a relational or flat file source or target. Use the relational or flat file source or target to define the columns in the web service source and target definitions.

When you generate a mapping from a relational or flat file source or target, the web service mapping generated contains the following objects:

- Web service source definition
- Source qualifier
- Web service target definition

The Designer links the ports from the source instance through the target instance. To complete the mapping, add the transformations and other mapping component necessary for the web service you want to create.

Note: When you generate a mapping from a relational or flat file source or target, create the web service source and target in the same process. When you run a workflow that contains a mapping with the web service source and target created at different times, the workflow can fail.

To generate a mapping from a relational or flat file source or target:

1. In the CDI-PC Designer, open the Mapping Designer.
2. Click Mappings > Create Web Service Mappings > Use Source/Target definitions.

   The procedure to generate a web service mapping from a relational or flat file source or target is the same as the procedure to create web service source and target definitions from relational or flat file sources or targets. For more information, see "Creating a Source or Target Definition" on page 41.
3. Save the mapping to the repository.

Generating a Mapping from a Transformation or Mapplet

You can generate a mapping from a reusable transformation or a mapplet. The Designer uses the ports in the transformation or mapplet to generate the web service source and target definitions.

Note: When you generate a mapping from a transformation or mapplet, create the web service source and target in the same process. When you run a workflow that contains a mapping with the web service source and target created at different times, the workflow can fail.

Generating a Mapping from a Reusable Transformation

The following table describes the types of transformation from which you can generate a web service mapping:

Transformation	Type	Group
Expression	Passive	Single
HTTP	Passive	One input and one output
Java	Active or passive	One input and one output
Lookup	Passive	Single
SQL	Active or passive	One input and one output
Stored Procedure	Passive	Single

The transformation you use to generate a web service mapping must be a reusable transformation. When you generate a web service mapping based on a transformation, the Designer lists the reusable transformations and shortcuts to reusable transformations available in the folder.

When you generate a web service mapping from a transformation, the Designer uses the ports in the transformation to define the columns for the source and target definitions. It then creates a mapping that contains a source with XML views that reflect the transformation input ports and a target definition with XML views that reflect the transformation output ports.

The web service mapping generated from a transformation contains the following objects:

- Web service source definition
- Source qualifier
- Transformation used to generate the mapping
- Web service target definition

The Designer links the ports from the source instance through the target instance.

Generating a Mapping from a Mapplet

You can generate a web service mapping from the following types of mapplets:

- Mapplets that contains one input transformation and one output transformation
- Mapplets that contains no active transformation

When you generate a web service mapping from a mapplet, the Designer lists the mapplets and shortcuts to mapplets that are permitted for the process.

When you generate a web service mapping from a mapplet, the Designer uses the ports in the mapplet to define the columns for the source and target definitions. It then creates a mapping that contains a source with XML views that reflect the mapplet input ports and a target definition with XML views that reflect the mapplet output ports.

The web service mapping generated from a mapplet contains the following objects:

- Web service source definition
- Source qualifier
- Mapplet used to generate the mapping
- Web service target definition

The Designer links the ports from the source instance through the target instance.

Generating a Mapping from a Reusable Transformation or a Mapplet

You use the same procedure to generate a web service mapping from a reusable transformation or from a mapplet.

To generate a web service mapping from a reusable transformation or mapplet:

1. In the Mapping Designer, click Mappings > Create Web Service Mapping > Use Transformation/Mapplet definitions.
2. Select the transformation or mapplet you want to use for the web service mapping.

   The Designer displays the list of input ports and the datatype, precision, and scale.

   You can specify whether the columns in the source and target definitions in the mapping occur once or multiple times.
3. If the columns occur multiple times, select Source and Target are Multiple Occurring Objects.

   This option indicates that the columns in the source and target are arrays. The columns as a group occur multiple times.
4. To add message and client ports to the source or target definition, click Add Message Ports.

   The Designer adds the message and client ports to the envelope view of the source and target definitions.
5. Click OK.

   The Designer creates the web service mapping and displays a message that the mapping was created successfully. It uses the transformation or mapplet name as the name of the source and target definitions prefixed with *m_*. It adds the suffixes *_input* to the source definition name and *_output* to the target definition name.

Editing a Target Instance in a Web Service Mapping

After you generate the web service mapping, you can edit the target instance in the mapping. When you edit the target instance in the Mapping Designer, you can edit properties that are not available in the Target Designer.

To edit the target definition in a web service mapping, double-click the target definition instance in the Mapping Designer.

You can edit the following transformation attributes on the Properties tab:

- Load scope
- Partial load recovery

Load Scope

The load scope attribute specifies the load scope for the target. The load scope in a web service target definition is similar to the transformation scope in a transformation.

You can set the load scope to the following values:

- Transaction. When you set the load scope to transaction, the Integration Service generates a response when it receives all data in the transaction. All groups in the target must receive data from the same transaction generator.
- All Input. When you set the load scope to All Input, the Integration Service generates a response after it receives all incoming data. Different groups in the target can receive data from different transaction generators. The Integration Service ignores commits when the load scope is All Input.

Partial Load Recovery

The partial load recovery attribute specifies how the target handles a previous partial load during recovery.

For a web service target, use the default value of None. You cannot specify recovery for a web service.

Attachments

You can configure CDI-PC web service workflows to use attachments in the following ways:

- Using a flat file or XML source or target as attachments to a SOAP message
- Using a WSDL with MIME attachments

Flat File or XML Source and Target Attachments

You can receive or send data as an attachment to a SOAP message request or response. The source or target can be a flat file or an XML document. For example, you periodically use FTP to access a flat file containing messages from a web service application. Instead of using FTP, you can create a web service workflow to receive the data from the flat file as an attachment to a SOAP request.

To receive data as attachment to a SOAP message request, use a flat file or XML source definition in the mapping. To use a flat file as a source for a web service, configure the reader to use a Web Services Provider Reader for Flat Files. Edit the web service session that runs the mapping. In the session properties, click the Mapping tab and select the source. Change the reader from Flat File Reader to Web Services Provider Reader for Flat Files.

To send data as attachment to a SOAP message response, use a flat file or XML target definition in the mapping. To use a flat file as a target for a web service, configure the writer to use a Web Services Provider Writer for Flat Files. Edit the web service session that runs the mapping. In the session properties, click the Mapping tab and select the target. Change the writer from Flat File Writer to Web Services Provider Writer for Flat Files.

WSDL Attachments

Based on the source and target definitions, you can receive and send an attachment as part of the SOAP request. The attachment must be a text file such as an XML document. You cannot attach binary documents such as JPEG, GIF, or PDF files. For example, you can extract an XML document from an Oracle database and pass it to a web service client as an attachment to a response message.

To use a binary file as a source, convert the file into hexbinary or base64binary before you pass it to the web service source. A hexbinary or base64binary files is treated as a text file. Similarly, you can convert the text file response generated by the web service target to a binary file.

The following table describes the attachment group ports in a web service definition:

Port Name	Description
FK_Att_*Name*	Generated foreign key pointing to PK_Message in the root group.
Att_Data_*Name*	Contains the attachment. You can view the MIME type for the attachment on the Attributes tab.
Att_Index_*Name*	Unique identifier for each attachment in the message.
Att_Type_*Name*	Type of attachment.

Rules and Guidelines for Using a WSDL with MIME Attachments

Use the following rules and guidelines when you work with attachments:

- A request or response can contain one attachment.
- The attachment must be a text file and use the UTF-8 code page or a code page that is a subset of the UTF-8 code page.
- To pass an attachment through requests or responses, you must connect all ports in the attachment group.
- If a definition in the mapping contains an attachment group, but you do not want to send or receive attachments, do not connect any of the ports in the group.
- If you receive messages from other sources, and each message contains an attachment, use a Sequence Generator transformation to generate a unique index for each attachment you send in a response.
- To send or receive an attachment, use a toolkit that supports MIME attachments to create the client application.

CHAPTER 8

Working with Web Service Workflows

This chapter includes the following topics:

- Working with Web Service Workflows Overview, 57
- Creating and Configuring a Web Service Workflow, 58
- Configuring the Web Services Provider Reader and Writer, 61
- Configuring Partitions for Web Service Sessions, 64
- Troubleshooting Web Service Workflows, 65

Working with Web Service Workflows Overview

Use the Workflow Manager to create a web service workflow. To create a web service workflow, enable the Web Service option for the workflow and then configure the web service properties.

When you create a session in a web service workflow, the session is called a web service session. You can include the following types of mappings in a web service session:

- Web service mapping
- Flat file mapping
- XML mapping

A web service session uses a Web Services Provider reader and writer. If a web service mapping contains a web service source and target, the session uses the Web Services Provider reader and writer by default. If a web service mapping contains a flat file or XML source or target, you must change the reader and writer type to the Web Services Provider reader or writer.

When a web service session contains XML or flat file sources or targets, the client application sends a request to the Web Services Hub as a MIME attachment to the SOAP message. To send or receive attachments, a client application must be created using a toolkit that supports MIME attachments.

When the Web Services Hub receives a SOAP message request to run a web service workflow, it passes the request to the Integration Service. After the Integration Service runs the web service request, it passes the response to the Web Services Hub. The Web Services Hub generates a SOAP message response and passes it back to the web service client.

You can set up multiple partitions in a session that contains web service source and target definitions. The Integration Service creates a connection to the Web Services Hub based on the number of sources, targets, and partitions in the session.

Note: Before you can run a web service workflow, you must create and configure a Web Services Hub in the Administrator tool and associate it with the repository that contains the web service workflow you want to run.

Creating and Configuring a Web Service Workflow

To create a web service workflow, configure a workflow to process a web service mapping and enable the Web Services option in the workflow properties. You can configure the web service to allow web service clients to run the workflow.

To create and configure a web service workflow, complete the following tasks:

- Create a web service workflow.
- Configure the web service.

Creating a Web Service Workflow

To create a web service workflow, enable the Web Services option for a workflow. Then configure the web service and add web service sessions to the workflow. A web service session is based on a web service mapping.

In most cases, a web service workflow contains one web service source for the input message and one web service target for the output message. The session can write to multiple fault views in a target. A one-way web service does not send a response and does not require a web service target.

Ensure that you specify an Integration Service when you create a web service workflow. Use the Browse Integration Service button to select from a list of available Integration Services.

After you create the web service workflow, you can add a session to run a web service mapping. You create and add a session to the web service workflow the same way you create and add a session to any workflow.

Note: Do not use the Workflow Wizard to create a web service workflow. You cannot select the Web Service option when you use the Workflow Wizard.

To create a web service workflow:

1. In the Workflow Manager, open the Workflow Designer and click Workflows > Create.
2. Enter the name for the workflow.
3. To select the Integration Service to run the workflow, click the Browse Integration Service button and select from the list.
4. Enable the Web Services option and click Config Service to configure the web service workflow.

   When you enable the Web Services option, the Configure Concurrent Execution option is enabled by default. The web service workflow configuration properties include settings for concurrent execution of the web service.
5. Configure the web service workflow properties as necessary.
6. Click OK.

Configuring the Web Service Workflow

When you configure a web service workflow, you can assign which Web Services Hub runs the web service workflow and configure the options for running and accessing the web service.

The following table describes the properties you can configure for a web service:

Property	Description
Service Name	Name of the web service. The Web Services Hub publishes this name when you check in the workflow and the service is visible. The default name is a concatenation of the repository name, folder name, and workflow name. This name must be unique.
Timeout (Seconds)	Maximum amount of time the Web Services Hub can take to process a request and generate a SOAP response before the request times out. If the Web Services Hub is unable to generate a response within the timeout period, it sends a fault message to the web service client and drops the connection. Default is 60 seconds. Set to 0 to disable the timeout period.
Service Time Threshold (Milliseconds)	Maximum amount of time the Web Services Hub can take to process requests before it starts another instance to process the next request. The service time period starts from the time the Web Services Hub receives a SOAP request to the time it generates a SOAP response. If the average time it takes the Web Services Hub to process a request exceeds the service time, the Web Services Hub starts a new instance of the web service to process new requests. For example, the service time is set to 1000 milliseconds. If the Web Services Hub cannot process requests within 1000 milliseconds, the Web Services Hub starts another instance of the web service to process the next SOAP request. Default is 1000. Note: To avoid a decline in performance, do not set the service time threshold to lower than 100 milliseconds.
Web Services Hubs	Web Services Hub Service to run the workflow. Click the Browse button to select one or more Web Services Hub Service to run the web service workflow. By default, the web service workflow can run on any Web Services Hub Service associated with the repository. **Note:** If you plan to start the workflow manually, select a Web Services Hub to run the workflow. Do not select Run on All Hubs. Before you start the web service workflow, verify that the Web Services Hub is enabled.
Maximum Run Count Per Hub	Maximum number of web service instances that can be started by a Web Services Hub. All instances of the web service workflow running on the Web Services Hub are included in the count, whether the instance is started dynamically or manually. The Web Services Hub cannot start another web service instance after the maximum is reached.
Protected	Requires authentication before the web service can be run. The Web Services Hub authenticates the request based on the user name token. You can choose to protect the service or make it public. Any CDI-PC user who can run a workflow can run a protected web service workflow using the Workflow Manager, *pmcmd*, or LMAPI. If a web service is not protected, any web service client can start the service without authentication. For more information, see "Adding Security to a Client Request" on page 30.

Property	Description
Visible	Makes the web service visible in the Web Services Hub Console. When you make the service visible, the Web Services Hub publishes the web service and the WSDL on the Web Services Hub Console. You can test the web service and view and download the WSDL from the Web Services Hub Console. If the service is not visible, the Web Services Hub does not publish the web service WSDL.
Runnable	Allows a web service client to start the workflow by sending a request to the Web Services Hub. If the web service workflow is runnable, a web service client request can start the workflow or run the web service while the workflow is running. If you want a web service client to start the workflow, schedule the workflow to run on demand. If the web service workflow is not runnable, a web service client can invoke the web service while the workflow is running, but cannot start the workflow. If disabled, you can start the workflow through the Workflow Manager, LMAPI, or *pmcmd*.

Concurrent Execution of Web Service Workflows

The Web Services Hub determines when to start a new instance of a web service workflow based on the availability of resources and values you set for the properties of the web service. It determines when to shut down an instance of a web service workflow based on values you set for the properties of the Web Services Provider Reader.

Starting a New Instance

The Web Services Hub determines when to start another instance of a web service workflow based on the current resource usage and the following properties of the web service workflow:

- **Service time threshold.** If the average time that it takes the Web Services Hub to process a web service exceeds the service time threshold, the Web Services Hub starts another instance of the web service.
- **Maximum run count per hub.** The Web Services Hub starts an instance of the web service until the number of instances reach the maximum run count for the hub. If the maximum run count is reached, the Web Services Hub does not start a new instance of the web service even if the average service time threshold.

Shutting Down an Instance

The Web Services Hub shuts down a web service workflow instance based on the current resource usage and the following properties of the Web Services Provider Reader:

- **Idle time.** If a workflow instance does not receive any request within the idle time period, the Web Services Hub shuts down the workflow instance.
- **Message count.** When the number of messages received by a workflow instance reaches the maximum number of messages the Integration Service is configured to read within a session, the Web Services Hub shuts down the workflow instance.
- **Reader time limit.** When the Integration Service reaches the maximum amount of time it can read input messages from the Web Services Hub, the Integration Service stops reading input messages from the Web Services Hub. The Web Services Hub shuts down the workflow instance.

If any of these properties reaches the threshold value configured for the workflow, the Web Services Hub shuts down the web service workflow instance.

Related Topics:

- "Configuring the Web Service Workflow" on page 58
- "Configuring the Web Services Provider Reader" on page 61

Configuring the Web Services Provider Reader and Writer

When you configure a web service session, you can configure the session reader and writer. By default, a web service session with web service sources and targets uses a Web Services Provider reader and writer.

If a web service session contains a flat file or XML source or target, you must configure the session to use the Web Services Provider reader or writer. The Web Services Hub sends requests and responses as MIME attachments to the SOAP message.

When you configure the reader for a web service session, you configure terminating conditions, such as idle time and message count.

When you configure the writer for a web service session, you configure caching information that the Integration Service uses to cache target data. You can also configure the output format for the target data.

Use the Workflow Manager to configure a web service session. In the Workflow Designer, edit the session of a web service workflow. To configure the Web Services Provider reader, click the Mapping tab and select a source. To configure the Web Services Provider writer, select a target.

Related Topics:

- "Attachments" on page 55

Configuring the Web Services Provider Reader

The properties you configure for a Web Services Provider reader depend on the source type used in the mapping.

The following table describes the source properties you configure for the web service session:

Property	Reader Type	Description
Idle Time	- Web Service - Web Services Provider Reader Flat File - Web Services Provider Reader XML File	Amount of time in seconds the Integration Service waits to receive messages before the it stops reading from the source and the Web Services Hub shuts down the workflow instance. The session stops when it meets the condition of this property. Default is 180.
Message Count	- Web Service - Web Services Provider Reader Flat File - Web Services Provider Reader XML File	The number of messages the Integration Service reads before the Web Services Hub shuts down the workflow instance. A value of -1 indicates an infinite number of messages. If the session uses flat file or XML targets, always configure the message count to 1. For more information, see "Configuring the Reader and Writer for XML and Flat File Sessions" on page 64. The session stops when it meets the condition of this property. Default is -1.
Reader Time Limit	- Web Service - Web Services Provider Reader Flat File - Web Services Provider Reader XML File	Amount of time in seconds that the Integration Service reads source messages from the Web Services Hub. For example, if you set the reader time limit to 10, the Integration Service stops reading from the Web Services Hub after 10 seconds. The session stops when it meets the condition of this property. Default is 0 and indicates an infinite period of time.
Treat Empty Content as Null	Web Services Provider Reader XML File	Treats empty strings as null values. By default, empty content is not null.
Recovery Cache Folder	n/a	This property is not used by the Web Services Provider.

Configuring the Web Services Provider Writer

When you configure session properties for a Web Services Provider writer, you configure cache size and cache directory.

The following table describes the target properties you configure for the web service session:

Property	Writer Type	Description
XML DateTime Format	Web Services Provider Writer XML File	Datetime format for the data passed to the service target. Precision to the nanosecond. Select from the following datetime formats: - Local Time. The time according to the Integration Service server time zone. - Local Time with Time Zone. The difference in hours between the Integration Service time zone and Greenwich Mean Time. - UTC. Greenwich Mean Time.
Null Content Representation	Web Services Provider Writer XML File	Determines how null content is represented in the target. Select from the following options: - No Tag. Do not output a tag. - Tag with Empty Content. Output just the tag. Default is No Tag.
Empty String Content Representation	Web Services Provider Writer XML File	Determines how an empty string is represented in the target. Select from the following options: - No Tag. Do not output a tag. - Tag with Empty Content. Output just the tag. Default is Tag with Empty Content.
Duplicate Group Row Handling	Web Services Provider Writer XML File	Determines how the Integration Service handles duplicate group rows during a session. Select from the following options: - First Row. The Integration Service passes the first duplicate row to the target. The Integration Service rejects rows with the same primary key that it processes after this row. - Last Row. The Integration Service passes the last duplicate row to the target. - Error. The 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. Default is Error.
Orphan Row Handling	Web Services Provider Writer XML File	Determines how the Integration Service handles orphan rows during a session. Select from the following options: - Ignore. The Integration Service ignores orphan rows. - Error. The session fails when the error count exceeds the error threshold.

Property	Writer Type	Description
Cache Size	- Web Service - Web Services Provider Writer XML File	Total size in bytes for the memory cache used by writer. It includes a primary key and a foreign key index cache for each group in the target instance and one data cache for all groups. The total cache requirement is the sum of the data cache and index cache requirements for each target group. Default is 10,000,000 bytes.
Cache Directory	- Web Service - Web Services Provider Writer XML File	Directory for the target cache files. Default is the $PMCacheDir service process variable.

Use the following rules and guidelines when you change the writer type to a Web Services Provider writer:

- When you change the writer type for a flat file target, the Integration Service does not cache the target messages.
- When you change the writer type for a flat file or an XML target, use the target as a web service output message, but not as a fault message.
- When you change the writer type for an XML target, you still configure XML writer properties.

Configuring the Reader and Writer for XML and Flat File Sessions

To create a web service session based on a mapping that contains XML or flat file sources and targets, set the reader or writer type to Web Services Provider reader or writer. To run a web service workflow with an XML or flat file reader, a client application sends a request to the Web Services Hub as a MIME attachment to the SOAP message. The Web Services Hub passes the SOAP message with the attachment to the Integration Service, which processes the attachment.

If the web service workflow is configured with an XML or flat file writer, the Integration Service generates a response and passes the response to the Web Services Hub. The Web Services Hub sends the response back to the web service client as a MIME attachment to a SOAP message.

Use the following rules and guidelines when you configure a request-response web service session with flat file or XML source or targets:

- Set the message count to 1 in the reader properties.
- Include one session in a workflow where you change the reader or writer type to Web Services Provider.
- If you change the reader or writer type to Web Services Provider reader or writer in the session properties, you must create a client application using a toolkit that supports MIME attachments.

Configuring Partitions for Web Service Sessions

When you set up multiple partitions in a session that contains web service source and target definitions, the Integration Service creates a connection to the Web Services Hub based on the number of sources, targets, and partitions in the session. For example, if you configure three partitions in a session that contains one source and one target, the Integration Service creates six connections to the Web Services Hub, three for the source and three for the target. The partitions allow for concurrent execution of web service requests.

When you run a multi-partitioned session, the Web Services Hub uses a source connection to pass a request to the Integration Service. The Integration Service uses a target connection to send a response to the Web Services Hub. The Web Services Hub and the Integration Service use the source and target connections in a round-robin fashion.

When you configure partitions for a web service mapping, you can configure pass-through partitions for web service sources and targets.

Troubleshooting Web Service Workflows

I am trying to run the Debugger against a web service session, but the session fails, and the session log contains an error message indicating that the workflow context is required to run the session.

If you want to debug a web service session, you must run the Debugger against the web service workflow. You cannot run the Debugger against a web service mapping or a reusable session without the workflow.

I updated the source WSDL and reimported my source and target definitions. The workflow is valid, but the service WSDL is not updated.

Changes to a mapping are not dynamically reflected in the Web Services Hub. To generate the WSDL to reflect the mapping changes, you need to edit and save the workflow. When you save the workflow, the Web Services Hub generates the WSDL for the service.

My web service workflow was valid in the Workflow Manager, but became invalid when I started the Web Services Hub.

After you start the Web Services Hub, it validates each web service workflow according to its own validation rules in addition to those of the Workflow Manager.

The Web Services Hub validates web service workflows according to the following rules:

- There can be no more than one web service source definition in the mapping.
- There can be no more than one web service target definition in the mapping.
- If there are no web service target definitions in the mapping, the Web Services Hub treats the web service as a one-way service.
- A Repository Service must be associated with the Web Services Hub.
- An Integration Service must be associated with the workflow.

See the Validate tab in the Workflow Manager for Web Services Hub error messages, and correct the problem indicated by the error message.

While trying to fetch a workflow on a Web Services Hub, I received error messages indicating that there is no Integration Service specified for the service workflow and the service workflow is invalid.

You must assign an Integration Service when you create a web service workflow. For more information, see "Configuring the Web Service Workflow" on page 58.

I sent a request to a web service workflow that is configured to run more than one instance on the Web Services Hub. After I sent the request, I stopped the web service workflow. I received a fault response.

The Web Services Hub periodically checks the status of a workflow. It generates a fault response when it sends a request to the workflow before it registers that the workflow is not running. If a workflow is configured to run more than one instance, the Web Services Hub starts another instance of the workflow. However, since the Web Services Hub does not cache requests, it cannot resend the request to the new instance of the workflow.

I made changes to a real-time web service workflow in a versioned repository. When I ran the workflow, the changes were not in effect.

When you modify a real-time web service workflow in a versioned repository, you must check in the workflow for the changes to take effect.

For example, you modify a real-time web service workflow to associate it with a different CDI-PC Integration Service. If you check in the changes, the Web Services Hub uses the new Integration Service to run the workflow. If you do not check in the changes, the Web Services Hub does not use the new Integration Service unless you restart the Web Services Hub.

APPENDIX A

Web Service Sample Client Applications

This appendix includes the following topics:

- Web Service Sample Client Applications Overview, 67
- Using the Real-time Web Services Sample Programs, 67
- Examples of Real-time Web Services, 70

Web Service Sample Client Applications Overview

Informatica ships sample client application programs that demonstrate how to use CDI-PC web services. The sample programs work with the CDI-PC real-time web services.

The web services sample programs are installed in the following directory:

```
/<CDI-PCInstallDir>/server/samples/WebServices
```

Before running the web services sample programs, create and enable a Web Services Hub on the CDI-PC domain. Use the Administrator tool to create, configure, and enable a Web Services Hub.

Using the Real-time Web Services Sample Programs

Before you use the real-time web services sample programs, CDI-PC must be installed and running. The CDI-PC domain must contain a Web Services Hub associated with a Repository Service.

The real-time web services sample programs are installed in the following directory:

```
/<CDI-PCInstallDir>/server/samples/WebServices
```

The real-time web services examples include the files to create the lookup tables and web service workflows to be used by the sample programs.

The following table lists files and directories in the /RealTimeWebServices directory:

Directory	Description
/samples/RealTimeWebServices/ ImportXML	Contains the web service workflows called by the real-time web services sample programs. To use the sample programs, import the XML files into a repository and set up the database connections for the SQL and Lookup transformations in the web service workflows.
/lib	Contains the library files needed to run the sample programs.
/samples/RealTimeWebServices/ SQLScripts /SINGLEROWLOOKUP	Contains the SQL scripts for creating the lookup tables used in the sample program for single row lookup. Run the SQL scripts to create the tables in a database you select.
/samples/RealTimeWebServices/ SQLScripts / MULTIPLEROWLOOKUP	Contains the SQL scripts for creating the lookup tables used in the sample program for multiple row lookup. Run the SQL scripts to create the tables in a database you select.
/samples/RealTimeWebServices/ Unprotected WebServices/axis/ <*SampleProgramDirectory*>	Contains the Java sample programs. The source file for each real-time web services sample program can be found in a separate directory. Each directory contains the batch and script files to compile and run the sample program and subfolders for the proxy classes used by the sample program.

To use the real-time web services examples, you must complete the following steps:

1. Create the database tables that the sample programs will use as lookup tables.
2. Import the mappings and web service workflows into the repository associated with the Web Services Hub.
3. Modify the database and datatypes for the SQL transformation in the m_CustomerLookup_MULTIPLEROW mapping.
4. Set up the database connection settings in the sample workflows.
5. Compile the real-time web services sample programs.
6. Run the real-time web services sample programs.

Step 1. Create the Lookup Tables

Use the SQL script files shipped with the batch web services sample programs to create the lookup tables on a relational database. You can create the lookup tables in the following databases:

- IBM DB2
- Informix
- Microsoft SQL Server
- Oracle
- Sybase
- Teradata

Note: If you create the lookup tables in Teradata, you must set the default mode of the database server to ANSI.

The following table describes the SQL scripts:

Script File Name	Description
CustomerLookup_SINGLEROW_<Database>.sql	Creates a customer table named SINGLEROWLOOKUP for use with the sample program for single row lookup.
CustomerLookup_MULTIPLEROW_<Database>.sql	Creates a customer table named MULTIPLEROWLOOKUP for use with the sample program for multiple row lookup.

Note: the database connection settings. After you import the sample workflows into a repository, you need to modify the database connection settings of the transformations in the workflows to match your database settings.

Step 2. Import the Mappings and Workflows

The real-time web services sample programs run the sample web service workflows. To use the sample programs, import the sample mappings and workflows into the repository associated with the Web Services Hub.

The following table describes the XML files:

Script File Name	Description
wf_CustomerLookup_SINGLEROW.XML	Contains a web service workflow with a Lookup transformation for use with the sample program for single row lookup.
wf_CustomerLookup_MULTIPLEROW.XML	Contains a web service workflow with an SQL transformation for use with the sample program for multiple row lookup.

Step 3. Modify the Database and Datatypes for the SQL Transformation

The web service example that demonstrates multiple row lookup uses an SQL transformation. The database you use determines the native datatypes available for the ports in an SQL transformation. You must configure the SQL transformation to use the appropriate database and set the ports to use the appropriate native datatype.

To modify the database and datatypes for the SQL transformation, complete the following steps:

1. In the CDI-PC Designer, open the m_CustomerLookup_MULTIPLEROW mapping in Mapping Designer and edit the sql_Customer transformation instance.
2. In the Edit Transformations window, go to the SQL Settings tab and set the value of the Database Type attribute to the database you are using for the example.
3. Go to the SQL Ports tab and verify that the datatypes of the ports are mapped to the correct native datatype.

   For most databases, the default native datatype mapping is correct. For Microsoft SQL Server and Sybase, map the string datatype to the varchar native datatype.
4. Save the changes to the m_CustomerLookup_MULTIPLEROW mapping.

   After you modify the mappings, refresh the workflows that run the mappings.

5. In the CDI-PC Workflow Manager, open the workflows that run the mappings and refresh the mappings.

Step 4. Modify the Database Connection Settings

The SQL and Lookup transformations in the imported workflows must be able to connect to the sample lookup tables that you created in Step 1.

The import process does not import the connection object for the transformations in the sample workflows. You must create a connection object and use it in the session.

To update the connection settings for the transformations, complete the following steps:

1. In the CDI-PC Workflow Manager, create a connection object to connect to the sample tables.
2. Edit the s_m_CustomerLookup_SINGLEROW session and update the relational connection information in the lkp_Customer transformation.

   Set the relational connection to the name of the new connection object. Save the session with the new settings.
3. Edit the s_m_CustomerLookup_MULTIPLEROW session and update the relational connection information in the sql_Customer transformation.

   Set the relational connection to the name of your connection object. Save the session with the new settings.

Step 5. Compile the Real-time Web Service Sample Programs

To compile the sample Java programs, go to the sample program directory and run the compile batch or script file. Run the batch or script file that matches the name of the sample program you want to compile.

For example, to compile Sample.java program in the /axis/CustomerLookup_SINGLEROW directory, go to the directory and run CompileSample.bat (Windows) or CompileSample.sh (UNIX).

The compile process creates a .class file for the sample program in the same directory.

Step 6. Run the Real-time Web Service Sample Programs

You must have Java version 1.5.0_11-b03 installed on the machine where you run the sample programs. The Web Services Hub must be running when you run a sample program.

To run the sample Java programs, go to the sample program directory and run the batch or script file for the sample program you want to run. For example, to run the Sample.java program in the /axis/CustomerLookup_MULTIPLEROW directory, go to the directory and run RunSample.bat (Windows) or RunSample.sh (UNIX).

Run the sample program with the required parameters.

Examples of Real-time Web Services

This section describes the sample programs for real-time web services. Each directory contains a sample program that demonstrates a different way to use real-time web services.

Multiple Row Lookup

The sample program in the /CustomerLookup_MULTIPLEROW directory demonstrates how a client application can run a web service workflow to perform a lookup and handle a response with multiple rows of data.

Sample.java

This sample program calls a CDI-PC web service workflow that looks up a customer ID in a database and prints out the customer information. The workflow uses an SQL transformation to retrieve multiple rows from the database.

Directory: /CustomerLookup_MULTIPLEROW

File to compile Java sample: CompileSample.bat or CompileSample.sh

File to run Java sample: RunSample.bat or RunSample.sh

The following table describes the parameters you use to run the Sample application:

Parameter	Description
Customer ID	ID for the customer to look up. Pass the customer ID as an integer.
EndPoint URL	URL where the web service can be found. Pass the endpoint URL as a string. The endpoint URL for a real-time web service can be found in the soap:address location element of the service element in the web service WSDL. The default endpoint URL for the sample web service is *http*://<*WSHHostName*>:<*WSHPort*>/wsh/services/ts/CustomerLookup_MULTIPLEROW. If the Web Services Hub is running on HTTPS, the endpoint URL starts with HTTPS.

Single Row Lookup

The sample program in the /CustomerLookup_SINGLEROW directory demonstrates how a client application can run a web service workflow to perform a lookup and handle a response with single row of data.

Sample.java

This sample program calls a CDI-PC web service workflow that looks up a customer ID in a database and prints out the customer information. The mapping uses a Lookup transformation to retrieve one row from the database.

Directory: /CustomerLookup_SINGLEROW

File to compile Java samples: CompileSample.bat or CompileSample.sh

File to run Java sample: RunSample.bat or RunSample.sh

The following table describes the parameters you use to run the Sample application:

Parameter	Description
Customer ID	ID for the customer to look up. Pass the customer ID as an integer.
EndPoint URL	URL where the web service can be found. Pass the endpoint URL as a string. The endpoint URL for a real-time web service can be found in the soap:address location element of the service element in the Web service WSDL. The default endpoint URL for the sample web service is *http*://<*WSHHostName*>:<*WSHPort*>/wsh/services/ts/CustomerLookup_SINGLEROW. If the Web Services Hub is running on HTTPS, the endpoint URL starts with HTTPS.

APPENDIX B

Configure the Web Browser

This appendix includes the following topic:

- Configure the Web Browser, 73

Configure the Web Browser

You can use Microsoft Internet Explorer or Google Chrome to launch the Web Services Hub Console in the Informatica platform.

To run the Web Services Hub Console, configure the following options in your browser:

Scripting and ActiveX

Enable the following controls on Microsoft Internet Explorer:

- Active scripting
- Allow programmatic clipboard access
- Run ActiveX controls and plug-ins
- Script ActiveX controls marked safe for scripting

To configure the controls, click **Tools** > **Internet options** > **Security** > **Custom level**.

Trusted sites

If the Informatica domain runs on a network with Kerberos authentication, you must configure the browser to allow access to the Informatica web applications. In Microsoft Internet Explorer, Microsoft Edge, and Google Chrome, add the URL of the Informatica web application to the list of trusted sites. In Safari, add the certificate of the Informatica web application to the keychain. If you are using Chrome version 86.0.42x or later on Windows, you must also set the `AuthServerWhitelist` and `AuthNegotiateDelegateWhitelist` policies.

INDEX

A

attachments
  flat file mappings 55
  SOAP messages 57
  WSDL 56
  XML mappings 55

C

concurrent execution
  shutting down web service instances 60
  starting web service instances 60
configuring
  web service provider reader 61
  web service provider writer 62
Created element
  user name token security 33

D

digested password
  example 33
  web service security 30, 33

F

flat files
  mappings with attachments 55

H

hashed password
  example 32
  web service security 30, 31

M

mappings
  flat or XML with attachments 55
  one-way 50
  request-response 50
  staged 51
  types of web service mappings 50
  WSDL with attachment 56
  XML with attachments 55
message ports
  configuring 39

N

nonce
  user name token security 33

O

OASIS
  web service security standard 30
one-way mappings
  description 50

P

partitions
  web service sessions 64
passwords
  digested 30, 33
  hashed 30, 31
  plain text 30, 31
plain text password
  web service security 30, 31

R

reader
  configuring web service session 61
real-time web services
  description 12
  sample programs 67
request-response mappings
  description 50
  using a SQL transformation 50

S

sample programs
  real-time web services 67
SOAP
  attachments 57
SQL transformation
  request-response mappings 50
staged mapping
  description 51

U

user credential
  web service security 30
user name token
  Created element 33

user name token *(continued)*
nonce 33
web service security 30
UsernameToken element
web service security 30

W

web service instances
shutting down 60
starting 60
web service provider reader
configuring 61
web service provider writer
configuring 62
web service security
OASIS standard 30
user credential 30
web service security *(continued)*
user name token 30
web service source
configuring message ports 39
web service targets
configuring message ports 39
web service workflows
creating 58
troubleshooting 65
web services
Real-time 12
types of mappings 50
Web Services Hub
description 11
Web Services Provider
architecture 12
description 11
workflows
web service 58
writer
configuring web service session 62