Informatica®
10.1.0

Web Services Guide

Informatica Web Services Guide
10.1.0
June 2016

© Copyright Informatica LLC 1998, 2019

This software and documentation contain proprietary information of Informatica LLC and are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright law. Reverse engineering of the software is prohibited. No part of this document may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording or otherwise) without prior consent of Informatica LLC. This Software may be protected by U.S. and/or international Patents and other Patents Pending.

Use, duplication, or disclosure of the Software by the U.S. Government is subject to the restrictions set forth in the applicable software license agreement and as provided in DFARS 227.7202-1(a) and 227.7702-3(a) (1995), DFARS 252.227-7013©(1)(ii) (OCT 1988), FAR 12.212(a) (1995), FAR 52.227-19, or FAR 52.227-14 (ALT III), as applicable.

The information in this product or documentation is subject to change without notice. If you find any problems in this product or documentation, please report them to us in writing.

Informatica, Informatica Platform, Informatica Data Services, PowerCenter, PowerCenterRT, PowerCenter Connect, PowerCenter Data Analyzer, PowerExchange, PowerMart, Metadata Manager, Informatica Data Quality, Informatica Data Explorer, Informatica B2B Data Transformation, Informatica B2B Data Exchange Informatica On Demand, Informatica Identity Resolution, Informatica Application Information Lifecycle Management, Informatica Complex Event Processing, Ultra Messaging, Informatica Master Data Management, and Live Data Map are trademarks or registered trademarks of Informatica LLC in the United States and in jurisdictions throughout the world. All other company and product names may be trade names or trademarks of their respective owners.

Portions of this software and/or documentation are subject to copyright held by third parties, including without limitation: Copyright DataDirect Technologies. All rights reserved. Copyright © Sun Microsystems. All rights reserved. Copyright © RSA Security Inc. All Rights Reserved. Copyright © Ordinal Technology Corp. All rights reserved. Copyright © Aandacht c.v. All rights reserved. Copyright Genivia, Inc. All rights reserved. Copyright Isomorphic Software. All rights reserved. Copyright © Meta Integration Technology, Inc. All rights reserved. Copyright © Intalio. All rights reserved. Copyright © Oracle. All rights reserved. Copyright © Adobe Systems Incorporated. All rights reserved. Copyright © DataArt, Inc. All rights reserved. Copyright © ComponentSource. All rights reserved. Copyright © Microsoft Corporation. All rights reserved. Copyright © Rogue Wave Software, Inc. All rights reserved. Copyright © Teradata Corporation. All rights reserved. Copyright © Yahoo! Inc. All rights reserved. Copyright © Glyph & Cog, LLC. All rights reserved. Copyright © Thinkmap, Inc. All rights reserved. Copyright © Clearpace Software Limited. All rights reserved. Copyright © Information Builders, Inc. All rights reserved. Copyright © OSS Nokalva, Inc. All rights reserved. Copyright Edifecs, Inc. All rights reserved. Copyright Cleo Communications, Inc. All rights reserved. Copyright © International Organization for Standardization 1986. All rights reserved. Copyright © ej-technologies GmbH. All rights reserved. Copyright © Jaspersoft Corporation. All rights reserved. Copyright © International Business Machines Corporation. All rights reserved. Copyright © yWorks GmbH. All rights reserved. Copyright © Lucent Technologies. All rights reserved. Copyright (c) University of Toronto. All rights reserved. Copyright © Daniel Veillard. All rights reserved. Copyright © Unicode, Inc. Copyright IBM Corp. All rights reserved. Copyright © MicroQuill Software Publishing, Inc. All rights reserved. Copyright © PassMark Software Pty Ltd. All rights reserved. Copyright © LogiXML, Inc. All rights reserved. Copyright © 2003-2010 Lorenzi Davide, All rights reserved. Copyright © Red Hat, Inc. All rights reserved. Copyright © The Board of Trustees of the Leland Stanford Junior University. All rights reserved. Copyright © EMC Corporation. All rights reserved. Copyright © Flexera Software. All rights reserved. Copyright © Jinfonet Software. All rights reserved. Copyright © Apple Inc. All rights reserved. Copyright © Telerik Inc. All rights reserved. Copyright © BEA Systems. All rights reserved. Copyright © PDFlib GmbH. All rights reserved. Copyright © Orientation in Objects GmbH. All rights reserved. Copyright © Tanuki Software, Ltd. All rights reserved. Copyright © Ricebridge. All rights reserved. Copyright © Sencha, Inc. All rights reserved. Copyright © Scalable Systems, Inc. All rights reserved. Copyright © jQWidgets. All rights reserved. Copyright © Tableau Software, Inc. All rights reserved. Copyright© MaxMind, Inc. All Rights Reserved. Copyright © TMate Software s.r.o. All rights reserved. Copyright © MapR Technologies Inc. All rights reserved. Copyright © Amazon Corporate LLC. All rights reserved. Copyright © Highsoft. All rights reserved. Copyright © Python Software Foundation. All rights reserved. Copyright © BeOpen.com. All rights reserved. Copyright © CNRI. All rights reserved.

This product includes software developed by the Apache Software Foundation (http://www.apache.org/), and/or other software which is licensed under various versions of the Apache License (the "License"). You may obtain a copy of these Licenses at http://www.apache.org/licenses/. Unless required by applicable law or agreed to in writing, software distributed under these Licenses is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the Licenses for the specific language governing permissions and limitations under the Licenses.

This product includes software which was developed by Mozilla (http://www.mozilla.org/), software copyright The JBoss Group, LLC, all rights reserved; software copyright © 1999-2006 by Bruno Lowagie and Paulo Soares and other software which is licensed under various versions of the GNU Lesser General Public License Agreement, which may be found at http:// www.gnu.org/licenses/lgpl.html. The materials are provided free of charge by Informatica, "as-is", without warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose.

The product includes ACE(TM) and TAO(TM) software copyrighted by Douglas C. Schmidt and his research group at Washington University, University of California, Irvine, and Vanderbilt University, Copyright (©) 1993-2006, all rights reserved.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (copyright The OpenSSL Project. All Rights Reserved) and redistribution of this software is subject to terms available at http://www.openssl.org and http://www.openssl.org/source/license.html.

This product includes Curl software which is Copyright 1996-2013, Daniel Stenberg, <daniel@haxx.se>. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at http://curl.haxx.se/docs/copyright.html. Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

The product includes software copyright 2001-2005 (©) MetaStuff, Ltd. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at http://www.dom4j.org/ license.html.

The product includes software copyright © 2004-2007, The Dojo Foundation. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at http://dojotoolkit.org/license.

This product includes ICU software which is copyright International Business Machines Corporation and others. All rights reserved. Permissions and limitations regarding this software are subject to terms available at http://source.icu-project.org/repos/icu/icu/trunk/license.html.

This product includes software copyright © 1996-2006 Per Bothner. All rights reserved. Your right to use such materials is set forth in the license which may be found at http:// www.gnu.org/software/ kawa/Software-License.html.

This product includes OSSP UUID software which is Copyright © 2002 Ralf S. Engelschall, Copyright © 2002 The OSSP Project Copyright © 2002 Cable & Wireless Deutschland. Permissions and limitations regarding this software are subject to terms available at http://www.opensource.org/licenses/mit-license.php.

This product includes software developed by Boost (http://www.boost.org/) or under the Boost software license. Permissions and limitations regarding this software are subject to terms available at http:/ /www.boost.org/LICENSE_1_0.txt.

This product includes software copyright © 1997-2007 University of Cambridge. Permissions and limitations regarding this software are subject to terms available at http:// www.pcre.org/license.txt.

This product includes software copyright © 2007 The Eclipse Foundation. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at http:// www.eclipse.org/org/documents/epl-v10.php and at http://www.eclipse.org/org/documents/edl-v10.php.

This product includes software licensed under the terms at http://www.tcl.tk/software/tcltk/license.html, http://www.bosrup.com/web/overlib/?License, http://www.stlport.org/doc/ license.html, http://asm.ow2.org/license.html, http://www.cryptix.org/LICENSE.TXT, http://hsqldb.org/web/hsqlLicense.html, http://httpunit.sourceforge.net/doc/ license.html, http://jung.sourceforge.net/license.txt , http://www.gzip.org/zlib/zlib_license.html, http://www.openldap.org/software/release/license.html, http://www.libssh2.org, http://slf4j.org/license.html, http://www.sente.ch/software/OpenSourceLicense.html, http://fusesource.com/downloads/license-agreements/fuse-message-broker-v-5-3- license-agreement; http://antlr.org/license.html; http://aopalliance.sourceforge.net/; http://www.bouncycastle.org/licence.html; http://www.jgraph.com/jgraphdownload.html; http://www.jcraft.com/jsch/LICENSE.txt; http://jotm.objectweb.org/bsd_license.html; . http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231; http://www.slf4j.org/license.html; http://nanoxml.sourceforge.net/orig/copyright.html; http://www.json.org/license.html; http://forge.ow2.org/projects/javaservice/, http://www.postgresql.org/about/licence.html, http://www.sqlite.org/copyright.html, http://www.tcl.tk/software/tcltk/license.html, http://www.jaxen.org/faq.html, http://www.jdom.org/docs/faq.html, http://www.slf4j.org/license.html; http://www.iodbc.org/dataspace/iodbc/wiki/iODBC/License; http://www.keplerproject.org/md5/license.html; http://www.toedter.com/en/jcalendar/license.html; http://www.edankert.com/bounce/index.html; http://www.net-snmp.org/about/license.html; http://www.openmdx.org/#FAQ; http://www.php.net/license/3_01.txt; http://srp.stanford.edu/license.txt; http://www.schneier.com/blowfish.html; http://www.jmock.org/license.html; http://xsom.java.net; http://benalman.com/about/license/; https://github.com/CreateJS/EaselJS/blob/master/src/easeljs/display/Bitmap.js; http://www.h2database.com/html/license.html#summary; http://jsoncpp.sourceforge.net/LICENSE; http://jdbc.postgresql.org/license.html; http://protobuf.googlecode.com/svn/trunk/src/google/protobuf/descriptor.proto; https://github.com/rantav/hector/blob/master/LICENSE; http://web.mit.edu/Kerberos/krb5-current/doc/mitK5license.html; http://jibx.sourceforge.net/jibx-license.html; https://github.com/lyokato/libgeohash/blob/master/LICENSE; https://github.com/hjiang/jsonxx/blob/master/LICENSE; https://code.google.com/p/lz4/; https://github.com/jedisct1/libsodium/blob/master/LICENSE; http://one-jar.sourceforge.net/index.php?page=documents&file=license; https://github.com/EsotericSoftware/kryo/blob/master/license.txt; http://www.scala-lang.org/license.html; https://github.com/tinkerpop/blueprints/blob/master/LICENSE.txt; http://gee.cs.oswego.edu/dl/classes/EDU/oswego/cs/dl/util/concurrent/intro.html; https://aws.amazon.com/asl/; https://github.com/twbs/bootstrap/blob/master/LICENSE; https://sourceforge.net/p/xmlunit/code/HEAD/tree/trunk/LICENSE.txt; https://github.com/documentcloud/underscore-contrib/blob/master/LICENSE, and https://github.com/apache/hbase/blob/master/LICENSE.txt.

This product includes software licensed under the Academic Free License (http://www.opensource.org/licenses/afl-3.0.php), the Common Development and Distribution License (http://www.opensource.org/licenses/cddl1.php) the Common Public License (http://www.opensource.org/licenses/cpl1.0.php), the Sun Binary Code License Agreement Supplemental License Terms, the BSD License (http:// www.opensource.org/licenses/bsd-license.php), the new BSD License (http://opensource.org/licenses/BSD-3-Clause), the MIT License (http://www.opensource.org/licenses/mit-license.php), the Artistic License (http://www.opensource.org/licenses/artistic-license-1.0) and the Initial Developer's Public License Version 1.0 (http://www.firebirdsql.org/en/initial-developer-s-public-license-version-1-0/).

This product includes software copyright © 2003-2006 Joe Walnes, 2006-2007 XStream Committers. All rights reserved. Permissions and limitations regarding this software are subject to terms available at http://xstream.codehaus.org/license.html. This product includes software developed by the Indiana University Extreme! Lab. For further information please visit http://www.extreme.indiana.edu/.

This product includes software Copyright (c) 2013 Frank Balluffi and Markus Moeller. All rights reserved. Permissions and limitations regarding this software are subject to terms of the MIT license.

See patents at https://www.informatica.com/legal/patents.html.

DISCLAIMER: Informatica LLC provides this documentation "as is" without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of noninfringement, merchantability, or use for a particular purpose. Informatica LLC does not warrant that this software or documentation is error free. The information provided in this software or documentation may include technical inaccuracies or typographical errors. The information in this software and documentation is subject to change at any time without notice.

NOTICES

This Informatica product (the "Software") includes certain drivers (the "DataDirect Drivers") from DataDirect Technologies, an operating company of Progress Software Corporation ("DataDirect") which are subject to the following terms and conditions:

1. THE DATADIRECT DRIVERS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT.
2. IN NO EVENT WILL DATADIRECT OR ITS THIRD PARTY SUPPLIERS BE LIABLE TO THE END-USER CUSTOMER FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR OTHER DAMAGES ARISING OUT OF THE USE OF THE ODBC DRIVERS, WHETHER OR NOT INFORMED OF THE POSSIBILITIES OF DAMAGES IN ADVANCE. THESE LIMITATIONS APPLY TO ALL CAUSES OF ACTION, INCLUDING, WITHOUT LIMITATION, BREACH OF CONTRACT, BREACH OF WARRANTY, NEGLIGENCE, STRICT LIABILITY, MISREPRESENTATION AND OTHER TORTS.

Publication Date: 2019-04-29

Table of Contents

Preface 10

Informatica Resources. 10

Informatica Network. 10

Informatica Knowledge Base. 10

Informatica Documentation. 10

Informatica Product Availability Matrixes. 11

Informatica Velocity. 11

Informatica Marketplace. 11

Informatica Global Customer Support. 11

Chapter 1: Web Services 12

Web Services Overview. 12

Web Services Process. 12

Web Service Components. 13

Operations. 13

WSDL 14

SOAP. 14

Developing SOAP Web Services. 15

Web Services Examples. 15

Chapter 2: WSDL Data Object 17

WSDL Data Object Overview. 17

WSDL Data Object Overview View. 18

WSDL Data Object Advanced View. 18

Importing a WSDL Data Object. 18

WSDL Synchronization. 19

Synchronizing a WSDL Data Object. 19

Certificate Management. 20

Informatica Developer Certificate Properties. 20

Adding Certificates to Informatica Developer. 20

Chapter 3: Schema Object 21

Schema Object Overview. 21

Schema Object Overview View. 21

Schema Files. 22

Schema Object Schema View. 22

Namespace Properties. 22

Element Properties. 23

Simple Type Properties. 25

Complex Type Properties 26

Attribute Properties. . . . 26
Schema Object Advanced View. . . . 27
Creating a Schema Object. . . . 28
Schema Updates. . . . 28
Schema Synchronization. . . . 29
Schema File Edits. . . . 29
Certificate Management. . . . 31
Informatica Developer Certificate Properties. . . . 32
Adding Certificates to Informatica Developer. . . . 32

Chapter 4: Web Service Creation. . . . 33

Web Service Creation Overview. . . . 33
Types and Elements. . . . 34
Web Service Overview View. . . . 34
Web Service WSDL View. . . . 36
Create a Web Service from a WSDL Data Object. . . . 37
Step 1. Create a Web Service from a WSDL Data Object. . . . 37
Step 2. Add an Operation to a Web Service. . . . 38
Associating a WSDL Data Object with a Web Service. . . . 38
Manual Web Service Creation. . . . 38
Step 1. Creating a Web Service Manually. . . . 39
Step 2. Creating an Operation. . . . 39
Step 3. Create an Element. . . . 43
Step 4. Create a Predefined Fault. . . . 44
Step 5. Create a Header. . . . 44

Chapter 5: Operation Mappings. . . . 45

Operation Mappings Overview. . . . 45
Operation Mapping General Tab. . . . 46
Operation Mapping Operation Tab. . . . 46
Operation Mapping Advanced Tab. . . . 46
Input Transformation. . . . 47
Input Transformation Ports Tab. . . . 47
Rules and Guidelines to Map the Operation Input to Ports. . . . 47
Configuring the Input Transformation. . . . 48
Output Transformation. . . . 49
Output Transformation Ports Tab. . . . 49
Output Transformation Advanced Tab. . . . 50
Rules and Guidelines to Map Ports to the Operation Output. . . . 50
Configuring the Output Transformation. . . . 50
Fault Transformation. . . . 52
Fault Transformation Ports Tab. . . . 53
Fault Transformation Advanced Tab. . . . 53

Rules and Guidelines to Map Ports to the Operation Fault. 53
Creating a Fault Transformation. 53
Configuring the Fault Transformation. 54
Fault Handling. 55
System-Defined Faults. 57
Predefined Faults. 57
Generic Faults. 58
Test Operation Mappings. 59
Testing Operation Mappings. 59
Customized View Options. 59

Chapter 6: Parsing Web Service SOAP Messages. 60

Parsing Web Service SOAP Message Overview. 60
Transformation User Interface. 61
Multiple-Occurring Output Configuration. 62
Normalized Relational Output. 62
Generated Keys. 62
Denormalized Relational Output. 63
Pivoted Relational Output. 64
Parsing anyType Elements. 64
Parsing Derived Types. 65
Parsing QName Elements. 66
Parsing Substitution Groups. 66
Parsing XML Constructs in SOAP Messages. 66
Choice Element. 67
List Element. 67
Union Element. 67

Chapter 7: Generating Web Service SOAP Messages. 68

Generating Web Service SOAP Messages Overview. 68
Transformation User Interface. 69
Input Ports Area. 69
Operation Area. 70
Port and Hierarchy Level Relationships 70
Keys. 71
Map Ports. 72
Map a Port 73
Map a Group. 73
Map Multiple Ports. 74
Pivoting Multiple-Occurring Ports 74
Map Denormalized Data. 75
Derived Types and Element Substitution. 76
Generating Derived Types. 77

Generating anyType Elements and Attributes. 77
Generating Substitution Groups. 78
Generating XML Constructs in SOAP Messages. 78
Choice Element. 78
List Element. 79
Union Element. 79

Chapter 8: Web Service Administration. 81

Web Service Administration Overview. 81
Web Service Properties Configuration 81
Web Service Properties. 82
Web Service Operation Properties. 83
Web Service Result Set Caching. 83
Web Service Security Management. 84
Web Service Permissions. 85
UserName Token in a SOAP Request. 85
Web Service Logs. 88
Web Service Trace Levels. 88
Web Service Monitoring. 88
Properties View for a Web Service. 89
Reports View for a Web Service. 89
Operations View for a Web Service. 89
Requests View for a Web Service. 90

Chapter 9: Web Service Consumer Transformation. 91

Web Service Consumer Transformation Overview. 91
SOAP Messages. 92
WSDL Files. 92
Operations. 92
Web Service Security. 93
WSDL Selection. 94
Web Service Consumer Transformation Ports. 94
HTTP Header Input Ports. 95
Other Input Ports. 95
Web Service Consumer Transformation Input Mapping. 96
Rules and Guidelines to Map Input Ports to Nodes. 97
Customize View Options. 97
Mapping Input Ports to the Operation Input. 97
Web Service Consumer Transformation Output Mapping. 99
Rules and Guidelines to Map Nodes to Output Ports. 100
Mapping the SOAP Message as XML 100
Customize View Options. 100
Mapping the Operation Output to Output Ports. 101

Web Service Consumer Transformation Advanced Properties. 102
Web Service Error Handling. 104
Message Compression 104
Concurrency. 105
Filter Optimizations. 106
Enabling Early Selection Optimization with the Web Service Consumer Transformation. 106
Push-Into Optimization with the Web Service Consumer Transformation. 106
Creating a Web Service Consumer Transformation. 108
Web Service Consumer Transformation Example. 110
Input File. 110
Logical Data Object Model. 110
Logical Data Object Mapping. 110
Web Service Consumer Transformation. 111

Chapter 10: REST Web Service Consumer Transformation. 113

REST Web Service Consumer Transformation Overview. 113
REST Web Service Consumer Transformation Process. 114
REST Web Service Consumer Transformation Configuration. 115
Message Configuration. 115
Resource Identification. 115
HTTP Methods. 116
HTTP Get Method. 117
HTTP Post Method. 117
HTTP Put Method. 118
HTTP Delete Method. 119
REST Web Service Consumer Transformation Ports. 119
Input Ports. 120
Output Ports. 120
Pass-through Ports. 120
Argument Ports. 120
URL Ports. 120
HTTP Header Ports. 121
Cookie Ports. 121
Output XML Ports. 121
Response Code Ports. 122
REST Web Service Consumer Transformation Input Mapping. 122
Rules and Guidelines to Map Input Ports to Elements. 122
Mapping Input Ports to the Method Input. 123
REST Web Service Consumer Transformation Output Mapping. 124
Rules and Guidelines to Map Elements to Output Ports. 124
Customize View Options. 125
Mapping the Method Output to Output Ports. 125
REST Web Service Consumer Transformation Advanced Properties. 126

REST Web Service Consumer Transformation Creation. 127
Creating a REST Web Service Consumer Transformation. 127

Appendix A: Datatype Compatibility. 129

Datatype Reference Overview. 129
XML and Transformation Datatypes. 129
Decimal. 131

Index. 132

Preface

The Informatica *Web Services Guide* is written for data quality and data services developers. This guide assumes that you have an understanding of web services concepts.

Informatica Resources

Informatica Network

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

As a member, you can:

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

Informatica Knowledge Base

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

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

Informatica Documentation

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

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

Informatica Product Availability Matrixes

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

Informatica Velocity

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

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

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

Informatica Marketplace

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

Informatica Global Customer Support

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

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

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

CHAPTER 1

Web Services

This chapter includes the following topics:

- Web Services Overview, 12
- Web Services Process, 12
- Web Service Components, 13
- Developing SOAP Web Services, 15
- Web Services Examples, 15

Web Services Overview

Informatica Data Services provides data integration functionality through a web service. You can create a web service in the Developer tool. A web service client can connect to a web service to access, transform, or deliver data. An external application or a Web Service Consumer transformation can connect to a web service as a web service client.

A web service integrates applications using open standards, such as SOAP, WSDL, and XML. SOAP is the communications protocol for web services. The web service client request and the web service response are SOAP messages. A WSDL is an XML schema that describes the protocols, formats, and signatures of the web service operations.

Web service operations include requests for information, requests to update data, or requests to perform tasks. For example, a web service client sends a SOAP request to run a web service operation called getCustomerOrders. The web service client passes a customer ID in the request. The web service retrieves the customer and the order information. The web service returns the information to the client in a SOAP response.

Web Services Process

Web services receive requests from web service clients. An external application or a Web Service Consumer transformation can connect to a web service as a web service client.

The following process describes how the Data Integration Service processes web service requests from web service clients:

1. The Data Integration Service receives a SOAP request from a web service client.

2. The Web Service Module of the Data Integration Service processes the SOAP request by running an operation mapping.
3. The Web Service Module of the Data Integration Service sends a SOAP response to the web service client.

The following process describes how a Web Service Consumer transformation sends a requests and receives a response from a web service:

1. The transformation generates a SOAP request and connects to the web service. The Web Service Consumer transformation connects to a web service with a web service connection object.
2. The transformation receives the SOAP response from the web service.
3. The transformation extracts data from the SOAP response and returns the data in output ports.

Web Service Components

The components of a web service define the purpose of the web service and how the web service client communicates with the web service.

A web service has the following components:

Operations

A web service can have one or more operations. Each operation corresponds to an action in the web service.

Web Services Description Language (WSDL)

A WSDL is an XML document that describes the protocols, formats, and signatures of the web service operations.

Simple Object Access Protocol (SOAP)

SOAP is the communications protocol for web services.

Operations

A web service contains an operation for each action supported by the web service.

For example, a web service can have an operation named getcustomerid that receives a customer name and responds with the customer details. The operation input includes an element for the customer name. The operation output includes elements for customer details based on the customer name.

When you define an operation in the Developer tool, you define the operation components. An operation has the following components:

Operation input and output

The operation input defines the elements in the SOAP request for the operation. The operation output defines the elements in a SOAP response for the operation.

The operation input and the operation output can contain a header. A header receives or sends data within the SOAP message. The header defines the elements in the header of a SOAP request or SOAP response.

Operation faults

An operation fault defines the message format for error messages that might be output as the result of the operation. You can define multiple operation faults for an operation.

You must configure an operation mapping for each operation. The operation input, operation output, and each operation fault correspond to a transformation in the operation mapping.

WSDL

A WSDL is an XML schema that describes the protocols, formats, and signatures of the web service operations.

A 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 elements of a WSDL contain a description of the operations to be performed on that data, so that the receiver of a message knows how to process it. The elements of a WSDL also contain a binding to a protocol or transport, so that the sender of a message knows how to send it.

You can view the WSDL of a web service in the Developer tool or in the Administrator tool. After you deploy a web service to a Data Integration Service, you can view the WSDL URL or you can download the WSDL to a file. When you access the WSDL URL that the Administrator tool displays, you can see the content of the WSDL.

SOAP

SOAP is the communications protocol for web services. It defines the format for web service request, response, and fault messages. The Data Integration Service can process SOAP 1.1 and SOAP 1.2 messages with document/literal encoding.

A SOAP message contains the following sections:

SOAP envelope

The envelope defines the framework of the message, the content of the message, and what should handle the message.

SOAP header

The header identifies the entity that sent the SOAP message. It includes authentication information. It also includes information about how to process the SOAP message.

SOAP body

The body is the container for the data that the client and web service provider pass between each other.

SOAP messages are XML. When a SOAP message contains multiple-occurring elements, the groups of elements form levels in the XML hierarchy. The groups are related when one level is nested within another.

A SOAP request message can contain hierarchical data. For example, the client sends a request to add customer orders to a sales database. The client passes two groups of data in a SOAP request message. One group contains a customer ID and name, and the other group contains order information. The order information occurs multiple times.

A SOAP response message can contain hierarchical data. For example, a web service client generates a SOAP request for customer orders. The web service returns an order header and multiple-occurring order detail elements in the SOAP response.

Developing SOAP Web Services

Develop a SOAP web service to provide an interface that a web service client can use to perform operations. A web service client can be an external web service client or a Web Service Consumer transformation. For example, a web service client can connect to a web service to view customer details based on the customer name or the customer ID.

Complete the following steps to develop a web service:

1. Create a web service.
   - Create a web service from a WSDL data object. Import a WSDL file to create a WSDL data object. The WSDL file defines the operation input, the operation output, and the operation faults for a web service.
   - Manually create a web service. Configure the operation input, the operation output, and the operation faults. You can use elements and types from a schema object to define the operation components. You can use reusable mapplets, reusable transformations, and reusable logical data objects to define the elements of the operation input and operation output for an operation.
2. Configure operation mappings.
   Configure how the Data Integration Service extracts data between the SOAP messages and the Input transformation and Output transformation ports. Also, configure the operation mapping logic and test each operation mapping.
3. Deploy the web service to a Data Integration Service.
   Add the web service to an application and deploy the application to the Data Integration Service. When you deploy an application that contains a web service that is already running on the Data Integration Service, the Data Integration Service appends a number to the service name of the web service.
4. Complete administration tasks for the web service.
   Configure the web service properties and security in the Administrator tool.

A SOAP web service client can connect to a SOAP web service that is running on a Data Integration Service. Web service clients use the content of the WSDL to connect to a web service. You can configure the Web Service Consumer transformation to connect to a web service with a web service connection object.

Web Services Examples

Create a web service to meet various business requirements. For example, you can create a web service to access customer data or you can create a web service to validate customer address data.

Access Customer Data

Hypostores customer service representatives want to access customer data from the Los Angeles and Boston offices over a network. The customer service representatives want to view customer details based on the customer name or the customer ID. The corporate policy requires that data accessed over a network must be secure.

The developer and administrator complete the following steps to provide access to the data required by customer service:

1. In the Developer tool, the developer creates a web service with the following operations:
   - getCustomerDetailsByName
     The operation input includes an element for the customer name. The operation output includes elements for the customer details based on the customer name.
   - getCustomerDetailsById
     The operation input includes an element for the customer ID. The operation output includes elements for customer details based on the customer ID.
2. The developer configures an operation mapping for each operation with the following components:
   - An Input transformation and an Output transformation.
   - A Lookup transformation that performs a lookup on a logical data object that defines a single view of customer data from the Los Angeles and Boston offices.
3. The developer deploys the web service to a Data Integration Service.
4. In the Administrator tool, the administrator configures the web service to use transport layer security and message layer security so that it can receive authorized requests using an HTTPS URL.
5. The administrator sends the WSDL URL to customer service so that they can connect to the web service.

Validate Customer Address Data

The Hypostores fulfillment department wants to validate address data before finalizing orders. The Address Validator transformation compares input address data with address reference data to determine the accuracy of input addresses and fix errors in those addresses.

The developer and administrator complete the following steps to provide address validation functionality to the fulfillment department:

1. In the Developer tool, the developer creates a mapplet with an Address Validator transformation that receives address data as input and returns validated address data as output.
2. The developer creates a web service and uses the mapplet to create the web service operation.
   You can use the **Create Web Service** wizard to create an operation from an reusable object.
3. The developer deploys the web service to a Data Integration Service.
4. The administrator sends the WSDL URL to the fulfillment department so that they can connect to the web service.

The web service accepts an address as input and return a validated address as output.

CHAPTER 2

WSDL Data Object

This chapter includes the following topics:

- WSDL Data Object Overview, 17
- WSDL Data Object Overview View, 18
- WSDL Data Object Advanced View, 18
- Importing a WSDL Data Object, 18
- WSDL Synchronization, 19
- Certificate Management, 20

WSDL Data Object Overview

A WSDL data object is a physical data object that uses a WSDL file as a source. You can use a WSDL data object to create a web service or a Web Service Consumer transformation. Import a WSDL file to create a WSDL data object.

After you import a WSDL data object, you can edit general and advanced properties in the **Overview** and **Advanced** views. The **WSDL** view displays the WSDL file content.

Consider the following guidelines when you import a WSDL:

- The WSDL file must be WSDL 1.1 compliant.
- The WSDL file must be valid.
- Operations that you want to include in a web service or a Web Service Consumer transformation must use Document/Literal encoding. The WSDL import fails if all operations in the WSDL file use an encoding type other than Document/Literal.
- The Developer tool must be able to access any schema that the WSDL file references.
- If a WSDL file contains a schema or has an external schema, the Developer tool creates an embedded schema within the WSDL data object.
- If a WSDL file imports another WSDL file, the Developer tool combines both WSDLs to create the WSDL data object.
- If a WSDL file defines multiple operations, the Developer tool includes all operations in the WSDL data object. When you create a web service from a WSDL data object, you can choose to include one or more operations.

WSDL Data Object Overview View

The WSDL data object **Overview** view displays general information about the WSDL and operations in the WSDL.

The following table describes the general properties that you configure for a WSDL data object:

Property	Description
Name	Name of the WSDL data object.
Description	Description of the WSDL data object.

The following table describes the columns for operations defined in the WSDL data object:

Property	Description
Operation	The location where the WSDL defines the message format and protocol for the operation.
Input	The WSDL message name associated with the operation input.
Output	The WSDL message name associated with the operation output.
Fault	The WSDL message name associated with the operation fault.

WSDL Data Object Advanced View

The WSDL data object **Advanced** view displays advanced properties for a WSDL data object.

The following table describes the advanced properties for a WSDL data object:

Property	Description
Connection	Default web service connection for a Web Service Consumer transformation.
File Location	Location where the WSDL file exists.

Importing a WSDL Data Object

You can import a WSDL data object from a WSDL file or a URI that points to the WSDL location. You can import a WSDL data object from a WSDL file that contains either a SOAP 1.1 or SOAP 1.2 binding operation or both.

1. Click **File** > **New** > **Data Object.**
2. Select **WSDL data object** and click **Next**.

The **New WSDL Data Object** dialog box appears.

3. Click **Browse** next to the **WSDL** option and enter the location of the WSDL. Then, click **OK**.

   When you enter the location of the WSDL, you can browse to the WSDL file or you can enter the URI to the WSDL.

   Note: If the URI contains non-English characters, the import might fail. Copy the URI to the address bar in any browser. Copy the location back from the browser. The Developer tool accepts the encoded URI from the browser.

4. Enter a name for the WSDL.
5. Click **Browse** next to the **Location** option to select the project or folder location where you want to import the WSDL data object.
6. Click **Next** to view the operations in the WSDL.
7. Click **Finish**.

   The data object appears under **Physical Data Object** in the project or folder in the **Object Explorer** view.

WSDL Synchronization

You can synchronize a WSDL data object when the WSDL files change. When you synchronize a WSDL data object, the Developer tool reimports the object metadata from the WSDL files.

You can use a WSDL data object to create a web service or a Web Service Consumer transformation. When you update a WSDL data object, the Developer tool updates the objects that reference the WSDL and marks them as changed when you open them. When the Developer tool compares the new WSDL with the old WSDL, it identifies WSDL components through the name attributes.

If no name attribute changes, the Developer tool updates the objects that reference the WSDL components. For example, you edit a WSDL file and change the type for simple element "CustID" from xs:string to xs:integer.

If a name attribute changes, the Developer tool marks the objects that reference the WSDL component as changed when you open them.

The Developer tool validates the WSDL files before it updates the WSDL data object. If the WSDL files contain errors, the Developer tool does not import the files.

Synchronizing a WSDL Data Object

Synchronize a WSDL data object when the WSDL files change.

1. Right-click the WSDL data object in the **Object Explorer** view, and select **Synchronize**.

   The **Synchronize WSDL Data Object** dialog box appears.

2. Click **Browse** next to the **WSDL** field, and enter the location of the WSDL. Then, click **OK**.

   When you enter the location of the WSDL, you can browse to the WSDL file or you can enter the URI to the WSDL.

   Note: If the URI contains non-English characters, the import might fail. Copy the URI to the address bar in any browser. Copy the location back from the browser. The Developer tool accepts the encoded URI from the browser.

3. Verify the WSDL name and location.

4. Click **Next** to view the operations in the WSDL.
5. Click **Finish**.

   The Developer tool updates the objects that reference the WSDL and marks them as changed when you open them.

Certificate Management

The Developer tool must use a certificate to import WSDL data objects and schema objects from a URL that requires client authentication.

By default, the Developer tool imports objects from URLs that require client authentication when the server that hosts the URL uses a trusted certificate. When the server that hosts the URL uses an untrusted certificate, add the untrusted certificate to the Developer tool. If you do not add the untrusted certificate to the Developer tool, the Developer tool cannot import the object. Request the certificate file and password from the server administrator for the URL that you want import objects from.

The certificates that you add to the Developer tool apply to imports that you perform on the Developer tool machine. The Developer tool does not store certificates in the Model repository.

Informatica Developer Certificate Properties

Add certificates to the Developer tool when you want to import objects from a URL that requires client authentication with an untrusted certificate.

The following table describes the certificate properties:

Property	Description
Host Name	Name of the server that hosts the URL.
Port Number	Port number of the URL.
Certificate File Path	Location of the client certificate file.
Password	Password for the client certificate file.

Adding Certificates to Informatica Developer

When you add a certificate, you configure the certificate properties that the Developer tool uses when you import objects from a URL that requires client authentication with an untrusted certificate.

1. Click **Window** > **Preferences**.
2. Select **Informatica** > **Web Services** > **Certificates**.
3. Click **Add**.
4. Configure the certificate properties.
5. Click **OK**.

CHAPTER 3

Schema Object

This chapter includes the following topics:

- Schema Object Overview, 21
- Schema Object Overview View, 21
- Schema Object Schema View, 22
- Schema Object Advanced View, 27
- Creating a Schema Object, 28
- Schema Updates, 28
- Certificate Management, 31

Schema Object Overview

A schema object is a hierarchical schema that you import to the Model repository. After you import the schema, you can view the schema components in the Developer tool. You can import an Avro, Parquet, XML, or JSON schema. The Developer tool converts the schema to an .xsd file in the Model repository.

When you create a SOAP web service, you can define the structure of the web service based on a hierarchical schema. When you create a web service without a WSDL, you can define the operations, the input, the output, and the fault signatures based on the types and elements that the schema defines.

When you import a schema you can edit general schema properties in the **Overview** view. Edit advanced properties in the **Advanced** view. View the schema file content in the **Schema** view.

Schema Object Overview View

Select the **Overview** view to update the schema name or schema description, view namespaces, and manage schema files.

The **Overview** view shows the name, description, and target namespace for the schema. You can edit the schema name and the description. The target namespace displays the namespace to which the schema components belong. If no target namespace appears, the schema components do not belong to a namespace.

The **Schema Locations** area lists the schema files and the namespaces. You can add multiple root .xsd files. If a schema file includes or imports other schema files, the Developer tool includes the child .xsd files in the schema.

Schema Files

You can add multiple root-level .xsd files to a schema object. You can also remove root-level .xsd files from a schema object.

When you add a schema file, the Developer tool imports all .xsd files that are imported by or included in the file you add. The Developer tool validates the files you add against the files that are part of the schema object. The Developer tool does not allow you to add a file if the file conflicts with a file that is part of the schema object.

For example, a schema object contains root schema file "BostonCust.xsd." You want to add root schema file "LACust.xsd" to the schema object. Both schema files have the same target namespace and define an element called "Customer." When you try to add schema file LACust.xsd to the schema object, the Developer tool prompts you to retain the BostonCust.xsd file or overwrite it with the LACust.xsd file.

You can remove any root-level schema file. If you remove a schema file, the Developer tool changes the element type of elements that were defined by the schema file to xs:string.

To add a schema file, select the **Overview** view, and click the **Add** button next to the **Schema Locations** list. Then, select the schema file. To remove a schema file, select the file and click the **Remove** button.

Schema Object Schema View

The **Schema** view shows an alphabetic list of the groups, elements, types, attribute groups, and attributes in the schema. When you select a group, element, type, attribute group, or attribute in the **Schema** view, properties display in the right panel. You can also view each .xsd file in the **Schema** view.

The **Schema** view provides a list of the namespaces and the .xsd files in the schema object.

You can perform the following actions in the **Schema** view:

- To view the list of schema constructs, expand the **Directives** folder. To view the namespace, prefix, and the location, select a schema construct from the list.
- To view the namespace prefix, generated prefix, and location, select a namespace. You can change the generated prefix.
- To view the schema object as an .xsd file, select **Source**. If the schema object includes other schemas, you can select which .xsd file to view.
- To view an alphabetic list of groups, elements, types, attribute groups, and attributes in each namespace of the schema, select **Design**. You can enter one or more characters in the **Name** field to filter the groups, elements, types, attribute groups, and attributes by name.
- To view the element properties, select a group, element, type, attribute group, or attribute. The Developer tool displays different fields in the right panel based on the object you select.

When you view types, you can see whether a type is derived from another type. The interface shows the parent type. The interface also shows whether the child element inherited values by restriction or extension.

Namespace Properties

The **Namespace** view shows the prefix and location for a selected namespace.

The namespace associated with each schema file differentiates between elements that come from different sources but have the same names. A Uniform Resource Identifier (URI) reference defines the location of the file that contains the elements and attribute names.

When you import a schema that contains more than one namespace, the Developer tool adds the namespaces to the schema object. When the schema file includes other schemas, the namespaces for those schemas are also included.

The Developer tool creates a generated prefix for each namespace. When the schema does not contain a prefix, the Developer tool generates the namespace prefix tns0 and increments the prefix number for each additional namespace prefix. The Developer tool reserves the namespace prefix xs. If you import a schema that contains the namespace prefix xs, the Developer tool creates the generated prefix xs1. The Developer tool increments the prefix number when the schema contains the generated prefix value.

For example, Customer_Orders.xsd has a namespace. The schema includes another schema, Customers.xsd. The Customers schema has a different namespace. The Developer tool assigns prefix tns0 to the Customer_Orders namespace and prefix tns1 to the Customers namespace.

To view the namespace location and prefix, select a namespace in the **Schema** view.

When you create a web service from more than one schema object, each namespace must have a unique prefix. You can modify the generated prefix for each namespace.

Element Properties

An element is a simple or a complex type. A complex type contains other types. When you select an element in the **Schema** view, the Developer tool lists the child elements and the properties in the right panel of the screen.

The following table describes element properties that appear when you select an element:

Property	Description
Name	The element name.
Description	Description of the type.
Type	The element type.

The following table describes the child element properties that appear when you select an element:

Property	Description
Name	The element name.
Type	The element type.
Minimum Occurs	The minimum number of times that the element can occur at one point in an instance.
Maximum Occurs	The maximum number of times that the element can occur at one point in an instance.
Description	Description of the element.

To view additional child element properties, click the double arrow in the Description column to expand the window.

The following table describes the additional child element properties that appear when you expand the Description column:

Property	Description
Fixed Value	A specific value for an element that does not change.
Nillable	The element can have nil values. A nil element has element tags but has no value and no content.
Abstract	The element is an abstract type. An instance must include types derived from that type. An abstract type is not a valid type without derived element types.
Minimum Value	The minimum value for an element in an instance.
Maximum Value	The maximum value for an element in an instance.
Minimum Length	The minimum length of an element. Length is in bytes, characters, or items based on the element type.
Maximum Length	The maximum length of an element. Length is in bytes, characters, or items based on the element type.
Enumeration	A list of all legal values for an element.
Pattern	An expression pattern that defines valid element values.

Advanced Element Properties

To view advanced properties for a element, select the element in the **Schema** view. Click **Advanced**.

The following table describes the element advanced properties:

Property	Description
Abstract	The element is an abstract type. A SOAP message must include types derived from that type. An abstract type is not a valid type without derived element types.
Block	Prevents a derived element from appearing in the hierarchy in place of this element. The block value can contain "#all" or a list that includes extension, restriction, or substitution.
Final	Prevents the schema from extending or restricting the simple type as a derived type.
Substitution Group	The name of an element to substitute with the element.
Nillible	The element can have nil values. A nil element has element tags but has no value and no content.

Simple Type Properties

A simple type element is an element that contains unstructured text. When you select a simple type element in the **Schema** view, information about the simple type element appears in the right panel.

The following table describes the properties you can view for a simple type:

Property	Description
Type	Name of the element.
Description	Description of the element.
Variety	Defines if the simple type is union, list, anyType, or atomic. An atomic element contains no other elements or attributes.
Member types	A list of the types in a UNION construct.
Item type	The element type.
Base	The base type of an atomic element, such as integer or string.
Minimum Length	The minimum length for an element. Length is in bytes, characters, or items based on the element type.
Maximum Length	The maximum length for an element. Length is in bytes, characters, or items based on the element type.
Collapse whitespace	Strips leading and trailing whitespace. Collapses multiple spaces to a single space.
Enumerations	Restrict the type to the list of legal values.
Patterns	Restrict the type to values defined by a pattern expression.

Simple Type Advanced Properties

To view advanced properties for a simple type, select the simple type in the **Schema** view. Click **Advanced**.

The advanced properties appear below the simple type properties.

The following table describes the advanced property for a simple type:

Property	Description
Final	Prevents the schema from extending or restricting the simple type as a derived type.

Complex Type Properties

A complex type is an element that contains other elements and attributes. A complex type contains elements that are simple or complex types. When you select a complex type in the **Schema** view, the Developer tool lists the child elements and the child element properties in the right panel of the screen.

The following table describes complex type properties:

Property	Description
Name	The type name.
Description	Description of the type.
Inherit from	Name of the parent type.
Inherit by	Restriction or extension. A complex type is derived from a parent type. The complex type might reduce the elements or attributes of the parent. Or, it might add elements and attributes.

To view properties of each element in a complex type, click the double arrow in the Description column to expand the window.

Complex Type Advanced Properties

To view advanced properties for a complex type, select the element in the **Schema** view. Click **Advanced**.

The following table describes the advanced properties for a complex element or type:

Property	Description
Abstract	The element is an abstract type. A SOAP message must include types derived from that type. An abstract type is not a valid type without derived element types.
Block	Prevents a derived element from appearing in the schema in place of this element. The block value can contain "#all" or a list that includes extension, restriction, or substitution.
Final	Prevents the schema from extending or restricting the simple type as a derived type.
Substitution Group	The name of an element to substitute with the element.
Nillible	The element can have nil values. A nil element has element tags but has no value and no content.

Attribute Properties

An attribute is a simple type. Elements and complex types contain attributes. Global attributes appear as part of the schema. When you select a global attribute in the **Schema** view, the Developer tool lists attribute properties and related type properties in the right panel of the screen.

The following table describes the attribute properties:

Property	Description
Name	The attribute name.

Property	Description
Description	Description of the attribute.
Type	The attribute type.
Value	The value of the attribute type. Indicates whether the value of the attribute type is fixed or has a default value. If no value is defined, the property displays default=0.

The following table describes the type properties:

Property	Description
Minimum Length	The minimum length of the type. Length is in bytes, characters, or items based on the type.
Maximum Length	The maximum length of the type. Length is in bytes, characters, or items based on the type.
Collapse Whitespace	Strips leading and trailing whitespace. Collapses multiple spaces to a single space.
Enumerations	Restrict the type to the list of legal values.
Patterns	Restrict the type to values defined by a pattern expression.

Schema Object Advanced View

View advanced properties for the schema object.

The following table describes advanced properties for a schema object:

Name	Value	Description
elementFormDefault	Qualified or Unqualified	Determines whether or not elements must have a namespace. The schema qualifies elements with a prefix or by a target namespace declaration. The unqualified value means that the elements do not need a namespace.
attributeFormDefault	Qualified or Unqualified	Determines whether or not locally declared attributes must have a namespace. The schema qualifies attributes with a prefix or by a target namespace declaration. The unqualified value means that the attributes do not need a namespace.
File location	Full path to the .xsd file	The location of the .xsd file when you imported it.

Creating a Schema Object

You can import a hierarchical schema file or sample file to create a schema object in the repository.

1. Select a project or folder in the **Object Explorer** view.
2. Click **File** > **New** > **Schema**.

   The **New Schema** dialog box appears.
3. To import a schema file, select **Create from schema**, and then browse to and select a hierarchical schema file.

   You can enter a URI or a location on the file system to browse. The Developer tool validates the schema you choose. Review validation messages. You can select an Avro, Parquet, JSON, or .xsd schema file.

   Note: If the URI contains non-English characters, the import might fail. Copy the URI to the address bar in any browser. Copy the location back from the browser. The Developer tool accepts the encoded URI from the browser.
4. To create a schema from a sample file, select **Create from a sample file**, and then browse to and select a hierarchical file.

   You can select an Avro, Parquet, JSON, or XML file.

   Note: If you select a file with a different extension that contains Avro, Parquet, JSON, or XML content, the wizard recognizes the file content.
5. Optionally, change the schema name.
6. Click **Next** to view a list of the elements and types in the schema.
7. Click **Finish** to import the schema.

   The schema appears under Schema Objects in the **Object Explorer** view. The Developer tool stores the schema as an .xsd file.
8. To change the generated prefix for a schema namespace, select the namespace in the **Object Explorer** view. Change the **Generated Prefix** property in the **Namespace** view.

Schema Updates

You can update a schema object when elements, attributes, types, or other schema components change. When you update a schema object, the Developer tool updates objects that use the schema.

You can update a schema object through the following methods:

Synchronize the schema.

Synchronize a schema object when you update the schema files outside the Developer tool. When you synchronize a schema object, the Developer tool reimports all of the schema .xsd files that contain changes.

Edit a schema file.

Edit a schema file when you want to update a file from within the Developer tool. When you edit a schema file, the Developer tool opens the file in the editor you use for .xsd files. You can open the file in a different editor or set a default editor for .xsd files in the Developer tool.

You can use a schema to define element types in a web service. When you update a schema that is included in the WSDL of a web service, the Developer tool updates the web service and marks the web service as

changed when you open it. When the Developer tool compares the new schema with the old schema, it identifies schema components through the name attributes.

If no name attribute changes, the Developer tool updates the web service with the schema changes. For example, you edit a schema file from within the Developer tool and change the maxOccurs attribute for element "Item" from 10 to unbounded. When you save the file, the Developer tool updates the maxOccurs attribute in any web service that references the Item element.

If a name attribute changes, the Developer tool marks the web service as changed when you open it. For example, you edit a schema outside the Developer tool and change the name of a complex element type from "Order" to "CustOrder." You then synchronize the schema. When you open a web service that references the element, the Developer tool marks the web service name in the editor with an asterisk to indicate that the web service contains changes. The Developer tool adds the CustOrder element type to the web service, but it does not remove the Order element type. Because the Developer tool can no longer determine the type for the Order element, it changes the element type to xs:string.

Schema Synchronization

You can synchronize a schema object when the schema components change. When you synchronize a schema object, the Developer tool reimports the object metadata from the schema files.

Use schema synchronization when you make complex changes to the schema object outside the Developer tool. For example, you might synchronize a schema after you perform the following actions:

- Make changes to multiple schema files.
- Add or remove schema files from the schema.
- Change import or include elements.

The Developer tool validates the schema files before it updates the schema object. If the schema files contain errors, the Developer tool does not import the files.

To synchronize a schema object, right-click the schema object in the **Object Explorer** view, and select **Synchronize**.

Schema File Edits

You can edit a schema file from within the Developer tool to update schema components.

Edit a schema file in the Developer tool to make minor updates to a small number of files. For example, you might make one of the following minor updates to a schema file:

- Change the minOccurs or maxOccurs attributes for an element.
- Add an attribute to a complex type.
- Change a simple object type.

When you edit a schema file, the Developer tool opens a temporary copy of the schema file in an editor. You can edit schema files with the system editor that you use for .xsd files, or you can select another editor. You can also set the Developer tool default editor for .xsd files. Save the temporary schema file after you edit it.

The Developer tool validates the temporary file before it updates the schema object. If the schema file contains errors or contains components that conflict with other schema files in the schema object, the Developer tool does not import the file.

Note: When you edit and save the temporary schema file, the Developer tool does not update the schema file that appears in the **Schema Locations** list. If you synchronize a schema object after you edit a schema file in the Developer tool, the synchronization operation overwrites your edits.

Setting a Default Schema File Editor

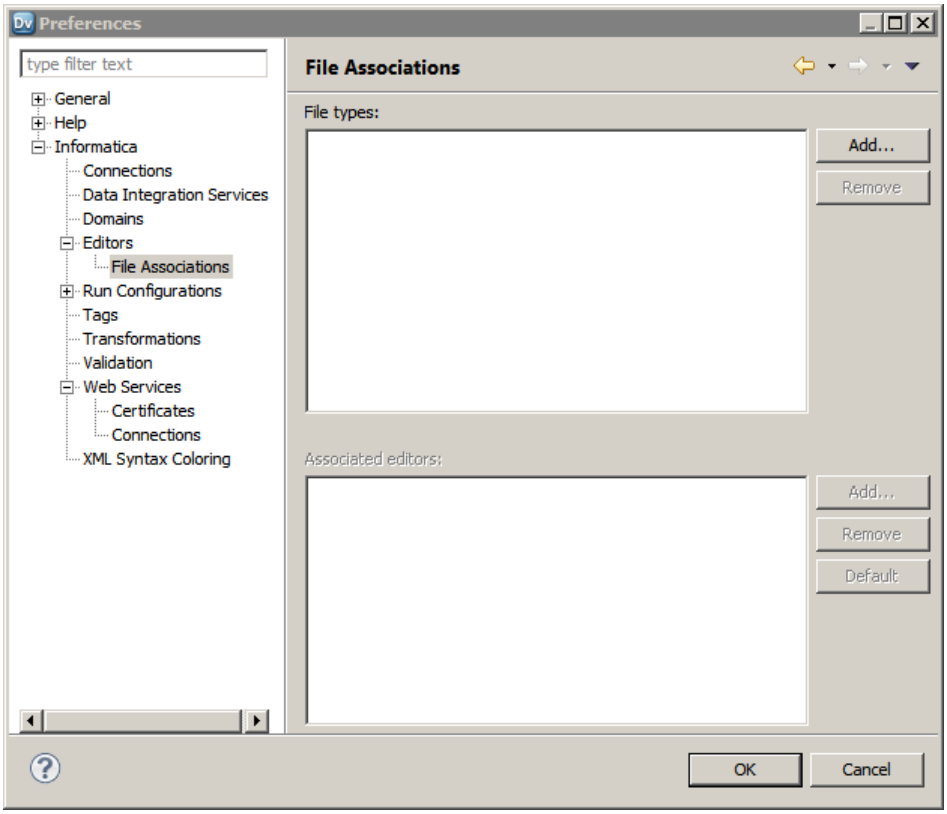
You can set the default editor that the Developer tool opens when you edit a schema file.

1. Click **Window** > **Preferences**.

   The **Preferences** dialog box appears.

2. Click **Editors** > **File Associations**.

   The **File Associations** page of the **Preferences** dialog box appears.



3. Click **Add** next to the **File types** area.

   The **Add File Type** dialog box appears.

4. Enter `.xsd` as the file type, and click **OK**.

5. Click **Add** next to the **Associated editors** area.

   The **Editor Selection** dialog box appears.

6. Select an editor from the list of editors or click **Browse** to select a different editor, and then click **OK**.

   The editor that you select appears in the **Associated editors** list.

7. Optionally, add other editors to the **Associated editors** list.

8. If you add multiple editors, you can change the default editor. Select an editor, and click **Default**.

9. Click **OK**.

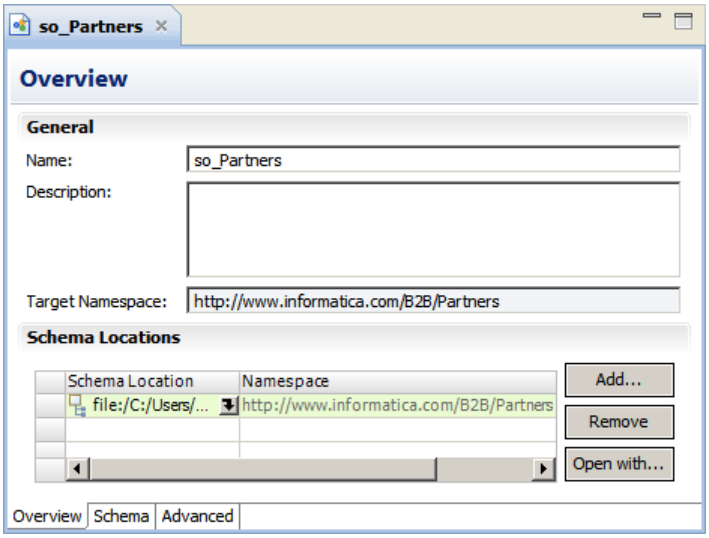
Editing a Schema File

You can edit any schema file in a schema object.

1. Open a schema object.

2. Select the **Overview** view.

   The **Overview** view of the schema object appears.



3. Select a schema file in the **Schema Locations** list.
4. Click **Open with**, and select one of the following options:

Option	Description
System Editor	The schema file opens in the editor that your operating system uses for .xsd files.
Default Editor	The schema file opens in the editor that you set as the default editor in the Developer tool. This option appears if you set a default editor.
Other	You select the editor in which to open the schema file.

   The Developer tool opens a temporary copy of the schema file.

5. Update the temporary schema file, save the changes, and close the editor.

   The Developer tool prompts you to update the schema object.

6. To update the schema object, click **Update Schema Object**.

   The Developer tool updates the schema file with the changes you made.

Certificate Management

The Developer tool must use a certificate to import WSDL data objects and schema objects from a URL that requires client authentication.

By default, the Developer tool imports objects from URLs that require client authentication when the server that hosts the URL uses a trusted certificate. When the server that hosts the URL uses an untrusted certificate, add the untrusted certificate to the Developer tool. If you do not add the untrusted certificate to the Developer tool, the Developer tool cannot import the object. Request the certificate file and password from the server administrator for the URL that you want import objects from.

The certificates that you add to the Developer tool apply to imports that you perform on the Developer tool machine. The Developer tool does not store certificates in the Model repository.

Informatica Developer Certificate Properties

Add certificates to the Developer tool when you want to import objects from a URL that requires client authentication with an untrusted certificate.

The following table describes the certificate properties:

Property	Description
Host Name	Name of the server that hosts the URL.
Port Number	Port number of the URL.
Certificate File Path	Location of the client certificate file.
Password	Password for the client certificate file.

Adding Certificates to Informatica Developer

When you add a certificate, you configure the certificate properties that the Developer tool uses when you import objects from a URL that requires client authentication with an untrusted certificate.

1. Click **Window** > **Preferences**.
2. Select **Informatica** > **Web Services** > **Certificates**.
3. Click **Add**.
4. Configure the certificate properties.
5. Click **OK**.

CHAPTER 4

Web Service Creation

This chapter includes the following topics:

- Web Service Creation Overview, 33
- Types and Elements, 34
- Web Service Overview View, 34
- Web Service WSDL View, 36
- Create a Web Service from a WSDL Data Object, 37
- Manual Web Service Creation, 38

Web Service Creation Overview

When you create a web service, you create a web service object in the repository. You can create a web service from a WSDL data object or you can manually create a web service. Each web service can have one or more operations.

When you create a web service from a WSDL data object, you choose the operations that you want to include in the web service from the WSDL data object. Each operation can use a SOAP 1.1 binding or a SOAP 1.2 binding but not both.

When you manually create a web service, you create operations and define the elements of the operation input, operation output, and operation faults for each operation. You can use a reusable object to define the elements of the operation input and operation output for an operation.

A web service object has an **Overview** view and a **WSDL** view. You can create and configure operations in the **Overview** view. You can view the WSDL file content in the **WSDL** view.

After you create a web service, configure the operation mapping for each operation. You can optionally add operations to the web service.

Types and Elements

When you create a web service from a WSDL data object, the WSDL data object defines the elements and element types for each operation. When you manually create an web service, you can define the elements or use types from schema objects to define the elements.

If operation components include anyType elements, any elements, anyAttribute attributes, derived type elements, or substitution groups, you must choose one or more types, elements, or attributes when you configure the operation mapping. For example, if the operation input includes an anyType element, choose the one or more types when you configure the Input transformation.

When you use a schema object to define element types, the Developer tool includes the schema object in the WSDL of the web service. When you delete the schema object or delete the link to the schema object in the web service WSDL, the Developer tool changes the element type of elements that were defined by the schema object to xs:string. You can update the element type to another type.

When you manually create a web service, the web service requires a unique generated prefix value for each namespace that it uses to define operation components. For example, if a web service uses schema object schemaA and schema object schemaB to define types, schemaA and schema B cannot have the same generated prefix value for any namespace.

Web Service Overview View

The web service **Overview** view displays general information about the web service and detailed information about the web service operations and operation components.

General Properties

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

Property	Description
Name	The web service object name.
Description	The description of the web service.
Namespace	The targetNamespace of the web service. If the web service is associated with a WSDL data object, this field is read-only.
Prefix	The prefix of the targetNamespace. If the web service is associated with a WSDL data object, this field is read-only.
WSDL Data Object	The WSDL data object associated with the web service. This property displays if the web service was created from a WSDL data object.
Service Name	The service name. Default is the name of the web service or the service name defined in the associated WSDL data object. If the web service is associated with a WSDL data object, this field is read-only.

Operations Area

When you select an operation on the left side of the Operations area, the details appear on the right side. The left side of the Operations area displays a tree view of the operation with the related input, output, and fault.

The following table describes the operation and the input and output properties that appear on the right side of the Operations area:

Properties	Description
Operation Name	The name of the operation.
Description	The description of the operation
Binding Type	The binding type of the operation.
Input	The WSDL message name associated with the operation input.
Output	The WSDL message name associated with the operation output.
Element Name	The name of the XSD element referred by the operation input or output.
Name	The child element of the element referred by the operation input or output.
Type	The element type.
Min. Occurs	The minimum number of times that the element can occur at one point in an XML instance.
Max. Occurs	The maximum number of times that the element can occur at one point in an XML instance.
Description	The description of the element.

To view additional child element properties, click the double arrow in the Description column to expand the window.

The following table describes the additional child element properties that appear when you expand the Description column:

Property	Description
Fixed Value	A specific value for an element that does not change.
Nillable	The element can have nil values. A nil element has element tags but has no value and no content.
Abstract	The element is an abstract type. An XML instance must include types derived from that type. An abstract type is not a valid type without derived element types.
Minimum Value	The minimum value for an element in an XML instance.
Maximum Value	The maximum value for an element in an XML instance.
Minimum Length	The minimum length of an element. Length is in bytes, characters, or items based on the element type.
Maximum Length	The maximum length of an element. Length is in bytes, characters, or items based on the element type.

Property	Description
Enumeration	A list of all legal values for an element.
Pattern	An expression pattern that defines valid element values.

Operation Mappings Area

The following table describes the columns for operation mappings:

Property	Description
Operation Mapping	The operation mapping name.
Operation	The operation name.
Input	The WSDL message name associated with the operation input.
Output	The WSDL message name associated with the operation output.
Fault	The WSDL message name associated with the operation fault.

Web Service WSDL View

You can preview the contents of the WSDL file on the **WSDL** view of a web service.

When you create a web service from a WSDL object, you can view the contents of the WSDL file.

When you manually create a web service, the Developer tool generates the contents of a WSDL file based on the configuration of the operation input, operation output, operation fault, and headers. You add elements to the schema of the WSDL or edit elements in the schema of the WSDL. You can also delete imported schema object from the WSDL.

Each operation input, operation output, operation fault, and header corresponds to a message in the WSDL. Operation faults and headers can share messages in a WSDL. When you use an element from the schema of the WSDL to create a fault or a header, the message name is the same as the element name.

You can view the WSDL design or the WSDL source. The WSDL design displays the hierarchical view of the WSDL contents. The WSDL source displays the content of the WSDL in an XML format.

To preview the WSDL in the Developer tool, select **Source** next to the **Show** field on the **WSDL** view of the web service.

Create a Web Service from a WSDL Data Object

When you create a SOAP web service from a WSDL data object, the WSDL defines one or more operations and the elements of each operation input, operation output, and operation fault.

You can add an operation if the WSDL has multiple operations that you can select from. You cannot create an operation that is not defined in the WSDL.

You must create a WSDL data object before you create a web service from a WSDL data object. The WSDL must be based on SOAP 1.1 or SOAP 1.2. You can use the **Create Web Service from a WSDL Data Object** wizard to create a WSDL data object before you create a web service from a WSDL data object.

When you create a web service from a WSDL data object, the web service is dependent on the WSDL data object. If you delete the WSDL data object, the operation mapping is not valid and you must associate a WSDL data object with the web service. The Developer tool removes the association between a web service and the WSDL data object when a WSDL data object is deleted from the repository.

Step 1. Create a Web Service from a WSDL Data Object

When you create a web service from a WSDL data object, the Developer tool uses information defined in the WSDL data object to create an operation mapping for each operation in the web service.

1. Select a project or folder in the **Object Explorer** view.
2. Click **File** > **New** > **Data Service**.

   The **New Data Service** dialog box appears.
3. Click **Web Service from a WSDL data object**, and then click **Next**.
4. To create a WSDL data object, click **New WSDL Data Object**.
   a. Click **Browse** next to the **Location** option to enter the location of the WSDL. Click **OK**.
   b. Enter a name for the WSDL.
   c. Click **Finish**.
5. Enter a name for the web service.
6. Click **Browse** next to the **WSDL Data Object** option to select the WSDL data object.
7. Click **Browse** next to the **Operation** option to select one or more operations that you want to include in the web service.

   The WSDL data object can contain multiple bindings of SOAP 1.1 and SOAP 1.2 formats. You can select a specific operation from only one of these bindings.
8. Click **Next**.
9. By default, the SOAP version of the operation you select will be displayed as the **Operation Type**.
10. Optionally, select the **Mapping Input** tab for each operation input to map data from the operation input to the output ports.

    You can also map the data when you configure the Input transformation.
11. Optionally, select the **Mapping Output** tab for each operation output to map data from the input ports to the operation output.

    You can also map the data when you configure the Output transformation.
12. Optionally, select the **Mapping Fault** tab for each operation fault to map data from the input ports to the operation fault.

    You can also map the data when you configure the Fault transformation.

13. Click **Finish**.

Step 2. Add an Operation to a Web Service

You can optionally add operations to a Web Service.

1. In the **Object Explorer** view, open a web service that is dependent on a WSDL data object.
2. Select the **Overview** view.
3. In the **Operations** section, select the **Operation mappings** area.
4. From the **Operations mappings** area, click **Choose**.

   You will see the **Select One or more Operations** dialog box.
5. Select the operation you want to add and click **OK**.

Associating a WSDL Data Object with a Web Service

To associate a WSDL data object with a web service, choose a WSDL data object and associate each operation mapping with an operation defined in the WSDL data object.

When you create a web service from a WSDL data object, the web service is dependent on the WSDL data object. If you delete the WSDL data object, the operation mapping is not valid and you must associate a WSDL data object with the web service.

1. In the **Object Explorer** view, open a web service that is dependent on a WSDL data object.
2. Select the **Overview** view.
3. Click **Browse** next to the **WSDL Data Object** option.

   The **Select a WSDL Data Object** dialog box appears.
4. Select the WSDL data object and click **OK**.
5. In the **Operation Mappings** area, associate a WSDL operation with each web service operation mapping.
   a. Right-click in the **Operation** column of an operation mapping row and click **Select an operation**.

      The **Select an Operation** dialog box appears.
   b. Select the operation and click **OK**.

Manual Web Service Creation

When you create a web service without a WSDL data object, you define the web service properties, operations, and operation components. Operation components include the operation input, the operation output, and operation faults.

When you manually create a web service, you can define one or more operations. When you manually create an operation, you can select the SOAP binding type and use elements and types from schema objects to define the operation elements. You can also use a mapplet, reusable transformation, logical data object, flat file data object, or relational data object to create an operation. When you create an operation from reusable object, you select the fields that you want to include in the operation from the object.

The Developer tool generates the contents of a WSDL file using the web service properties and the operations that you define. You can preview the contents of a WSDL file after you create the web service.

Step 1. Create a Web Service Manually

Use the **Create a Web Service** wizard to create a web service.

1. Select a project or folder in the **Object Explorer** view.
2. Click **File** > **New** > **Data Service.**

   The **New Data Service** dialog box appears.
3. Click **Web Service**.

   The **New Web Service** dialog box appears.
4. Enter a name for the web service.
5. Optionally, enter the namespace and the namespace prefix.
6. Click **Next**.

   The **New Web Service** dialog box appears.

You must create an operation after you create the web service.

Step 2. Create an Operation

When you manually create a web service, you can create an operation from the Create a Web Service wizard or the New Operation wizard.

The following table lists the options to create an operation:

Options to Create an Operation	Path
Use the Create a Web Service wizard.	Click **New** > **Data Service** > **Web Service**. From a reusable object or from an empty object. **Note:** A reusable object can be a physical data object, a mapplet, or a transformation.
Use the New Operation wizard.	Right-click a web service from the **Object Explorer** or **Outline** view and click **New > Operation**.
Use the New Operation wizard.	Create a web service and then select the web service **Overview** view. In the **Operations** area, click the arrow next to the **New** button and select **New Operation**.
Use the New Operation wizard.	Drag a reusable object into the **Operations** area of the **Overview** view.

Create an Operation from a Reusable Object

Use the **Create a Web Service** wizard to create a web service and create operations from reusable objects.

You can create operations that look up data in logical data objects, flat file data objects, and relational data objects. You can also create operations from a mapplet or reusable transformation. The mapplet or reusable transformation defines elements of the operation input and output. You cannot create an operation from a Web Service Consumer transformation.

After you complete the steps to create a web service, you can create an operation from a data object, a mapplet, or reusable transformation within the **Create a Web Service** wizard.

1. In the **New Web Service** dialog box, click the arrow next to the **New** button. Then, select **Operation** > **Create from a Reusable Object**.

   The **Select Reusable Object** dialog box appears.

   Note: You can also create an operation from a data object by opening a web service in the **Object Explorer** and selecting **Overview** > **Operations** area. Click the arrow next to the **New** button.
2. Select the object and click **OK**.
3. Optionally, enter the operation name and description.

   The Developer tool uses the name you provide for the operation to define the names for the operation input and the operation output.
4. Select the binding type of the operation.

   Default is SOAP 1.1. You cannot change the binding type of the operation after you have created and saved the operation.
5. Click the operation input to view and configure its properties.
   a. Select the operation input fields on the **Operation Input** tab.

      Note: If the object has more than one input group, select the input group and the operation input fields on the **Operation Input** tab.
   b. Optionally, configure the minimum and maximum number of occurrences for each element.
   c. Optionally, click the **Mapping Input** tab to configure how to map data from the operation input to the output ports.

      You can also map the data when you configure the Input transformation.
6. Click the operation output to display and configure its properties.
   a. Select the operation output fields on the **Operation Output** tab.

      Note: If the object has more than output group, select the output group and the operation output fields on the **Operation Output** tab.
   b. Optionally, configure the minimum number of occurrences for each element.
   c. Optionally, click the **Mapping Output** tab to configure how to map data from the input ports to the operation output.

      You can also map the data when you configure the Output transformation.
7. If the operation sends user-defined faults, click the arrow next to the **New** button and then click **Fault**.

   You can choose to create an element for the fault or select a reusable element.
8. Click each operation fault to display and configure its properties.

   You can click the **Mapping Fault** tab to configure how to map data from the input ports to the operation fault. You can also map the data when you configure the Fault transformation.
9. Repeat steps 1 through 8 to create and configure more operations.
10. Click **Finish**.

    The Developer tool creates an operation mapping for each operation based on how you configure each operation.

You can optionally create an element or a predefined fault.

Manually Create an Operation

Use the **Create Web Service** wizard to define operations and to define the operation input, operation output, and operation faults for each operation.

After you complete the steps to create a web service, you can create and configure the operation from within the **Create a Web Service** wizard.

1. In the **New Web Service** dialog box, click the arrow next to the **New** button. Then, select **Operation** > **Create as empty**.
   a. Enter a name for the operation.

      The Developer tool uses the operation name to define the names for the operation input and the operation output.
   b. Select the SOAP version for the operation as the **Binding Type**.
   c. To define fault messages, click the arrow next to the **New** button and then click **Fault.**

      You can choose to create each element for the fault message or select reusable elements. The wizard adds the elements to the detail element in the fault message.
2. Click the operation input to display and configure its properties.
   a. Click the **Operation Input** tab.
   b. To add elements, click the arrow next to the **New** button, and then click **Element**. To add child elements, select an element, click the arrow next to the **New** button and click **Child Element**.
   c. To specify a type for each element, click the selection button in the **Type** field. Choose an XSD type or a schema object type and click **OK**.

      Tip: Click the **Type** field and enter the first few characters of the type you want to select. A list of XSD types with names that start with the characters you enter will appear in a list.
   d. Configure the minimum and maximum number of occurrences for each element.
   e. Optionally, enter a description for each element.
   f. Optionally, click the **Mapping Input** tab to map data from the operation input to the output ports.

      You can also map the data when you configure the Input transformation.
3. Click each operation fault to display and configure its properties.

   You can click the **Mapping Fault** tab to configure how to map data from the input ports to the operation fault. You can also map the data when you configure the Fault transformation.
4. Click the operation output to display and configure its properties.
   a. Click the **Operation Output** tab.
   b. To add elements, click the arrow next to the **New** button, and then click **Element**. To add child elements, select an element, click the arrow next to the **New** button, and then click **Child Element**.
   c. To specify a type for each element, click the selection button in the **Type** field. Then, choose an XSD type or a schema object type and click **OK**.
   d. Configure the minimum and maximum number of occurrences for each element.
   e. Optionally, enter a description for each element.
   f. Optionally, click the **Mapping Output** tab to map data from the input ports to the operating output.

      You can also map the data when you configure the Output transformation.
5. Repeat steps 1 through 4 to add and configure more operations.
6. Click **Finish**.

   The Developer tool creates an operation mapping for each operation based on how you configure each operation.

You can optionally create an element or a predefined fault.

Creating an Operation from the New Operation Wizard

Use the New Operation wizard to create an operation after you have created a web service manually.

1. Choose a method to create an operation.
   - From the Object Explorer, right-click on a web service and select **New** > **Operation**.
   - From the Outline View, right-click on a web service and select **New** > **Operation**.
   - Open a web service in the **Object Explorer** and select **Overview** > **Operations** area. Click the arrow next to the **New** button.

   The **New Operation** dialog box apears.
2. Enter a name for the operation and optionally enter a description.

   The Developer tool uses the operation name to define the names for the operation input and the operation output.
3. Click **Next.**
4. Select the binding type of the operation.

   Default is SOAP 1.1. You cannot change the binding type of the operation after you have created and saved the operation.
5. Optionally, select the **Mapping Input** tab for each operation input to map data from the operation input to the output ports.

   You can also map the data when you configure the Input transformation.
6. Optionally, select the **Mapping Output** tab for each operation output to map data from the input ports to the operation output.

   You can also map the data when you configure the Output transformation.
7. Optionally, click the arrow next to the **New** button to create a Fault for the operation.

   You can choose to create each element for the fault message or select reusable elements. The wizard adds the elements to the detail element in the fault message. You can also map the data when you configure the Fault transformation.
8. Click **Finish**.

You can optionally create an element or a predefined fault.

Drag a Reusable Object

Drag a reusable object to the **Operations** area of the **Overview** view to create an operation from a reusable object.

1. Open a web service from the **Object Explorer** view.
2. From the **Overview** view, select the **Operations** area.
3. Drag a reusable object from the **Object Explorer** view to the **Operations** area in the **Overview** view.

   The **New Operation** dialog box appears.
4. Select the binding type of the operation.

   Default is SOAP 1.1. You cannot change the binding type of the operation after you have created and saved the operation.
5. Optionally, select the **Mapping Input** tab for each operation input to map data from the operation input to the output ports.

You can also map the data when you configure the Input transformation.

6. Optionally, select the **Mapping Output** tab for each operation output to map data from the input ports to the operation output.

   You can also map the data when you configure the Output transformation.

7. Optionally, click the arrow next to the **New** button to create a fault for the operation.

   You can also map the data when you configure the Fault transformation.

8. Click **Finish**.

You can optionally create an element or a predefined fault.

Step 3. Create an Element

You can optionally create an element on the **WSDL** view of a web service. You can create an operation fault or a header with elements you create in the **WSDL** view of the web service. An element can contain multiple elements and child elements.

Before you create an element, you must create a web service and define one or more operations from the **Create a Web Service** wizard.

1. Open the web service.
2. Select the **WSDL** view.
3. Show the **Design** of the WSDL. Then, in the Schema section select an element or the namespace entry above the elements.
4. Click the **New Element** button.
5. To add an element to the element, click the arrow next to the **New** button, and then click **Element**.
6. To add child elements to an element, select the element, click the arrow next to the **New** button, and then click **Child Element**.
7. Optionally, configure the type for each element.
   a. Click the **Selection** button in the **Type** field.
   b. Choose an XSD type or a Schema object type and click **OK**.
8. Optionally, configure the minimum and maximum number or occurrences for each element.
9. Optionally, enter a description for each element.

You can optionally create a predefined fault.

Change the Hierarchy Level of Elements

After you create an element, you can change the hierarchy level. Right-click in the **Operation Input** tab and **Operation Output** tab when you manually create an operation, to perform operations such as remove, move, or copy an element or child element.

You can also change the hierarchy of an element by dragging an element or a child element.

Step 4. Create a Predefined Fault

You can optionally create a fault to send user-defined errors within the SOAP response. When you create a predefined fault, the Developer tool adds a Fault transformation to the operation mapping.

Create a web service and define one or more operations. You can create a fault when you create an operation or you can add a fault to an operation. An operation can have multiple faults.

1. Open a web service.
2. In the **Outline** view, select the operation mapping.

   The operation mapping appears in the editor.
3. Click the **Operation** tab in the **Properties** view.
4. Click the arrow next to the **New** button.
5. Click **Fault**.
6. Choose to create an element or reuse an element.
   - Select **Create New Element** to create an element for the fault. You can create child elements and multiple-occurring elements.
   - Select **Reuse Existing Element** to reuse an element for the fault.

   The Developer tool adds elements to the detail element in the fault message hierarchy.

Step 5. Create a Header

You can optionally create a header to receive or send data within the header of the SOAP message. You can create a header for an operation input or operation output. You can use the HTTP POST method.

1. Open a web service.
2. Select the operation mapping in the **Outline** view.

   The operation mapping appears in the editor.
3. Click the **Operations** tab in the **Properties** view.
4. Choose to add a header to the operation input or the operation output.
   - Select the input to add the header to the operation input.
   - Select the output to add the header to the operation output.
5. Click the arrow next to the **New** button.
6. Click **Header** and then choose to create an element or reuse an element.
   - Select **Create New Element** to create a element for the header.
   - Select **Reuse Existing Element** to reuse an element for the header.

CHAPTER 5

Operation Mappings

This chapter includes the following topics:

- Operation Mappings Overview, 45
- Operation Mapping General Tab, 46
- Operation Mapping Operation Tab, 46
- Operation Mapping Advanced Tab, 46
- Input Transformation, 47
- Output Transformation, 49
- Fault Transformation, 52
- Fault Handling, 55
- Test Operation Mappings, 59
- Customized View Options, 59

Operation Mappings Overview

An operation mapping performs the web service operation for the web service client. An operation mapping can contain an Input transformation, an Output transformation, and multiple Fault transformations. The Input, Output, and Fault transformations process SOAP messages. The mapping can also contain other transformations that retrieve, transform, or update data based on the web service operation that the client requests.

After you create a web service in the Developer tool, configure an operation mapping for each operation in the web service. An operation mapping represents the logic for an operation.

When you configure the operation mapping, you define how the Data Integration Service processes the data that it receives in the SOAP request. The SOAP request can be of SOAP 1.1 or SOAP 1.2 format based on the binding type used by the binding operation associated with the operation mapping.

The Input transformation receives a SOAP request from a web service client and then returns data to transformations downstream in the mapping. The transformations perform the operation that the client requests.

The Output transformation receives data to return to the client. The Output transformation generates a SOAP response message to send to the client.

If an error occurs, the Data Integration Service generates a fault. The Data Integration Service returns user-defined faults from a Fault transformation. A user-defined fault consists of two types of faults: predefined fault and generic fault.

To configure the operation mapping, complete the following steps:

1. Configure the Input, Output, and Fault transformations.
2. Create and configure additional transformations to implement the operation logic.
3. Link ports.
4. Validate and save the mapping.

You can view and configure operation mapping properties on the **General** tab, **Operation** tab, and the **Advanced** tab in the operation mapping **Properties** view.

Operation Mapping General Tab

Configure the operation mapping name and description on the **General** tab of the operation mapping **Properties** view.

The **General** tab also displays the name of the operation that is associated with the operation mapping.

Operation Mapping Operation Tab

View or configure operation properties on the **Operation** tab of the operation mapping **Properties** view.

When you manually create a web service, you can use the **Operation** tab to define a fault or update the operation input or output. You can also add a header to an operation input or to an operation output.

Operation Mapping Advanced Tab

Configure operation mapping advanced properties on the **Advanced** tab of the **Properties** view. When you configure the advanced properties you can specify if you want the Data Integration Service to validate the XML of the SOAP request.

The following table describes the advanced properties for an operation mapping:

Property	Description
XML Schema Validation	Validates the SOAP request message at run time. Select **Error on Invalid XML** or **No Validation**. When the XML is not valid, the Data Integration Service returns a fault in the SOAP response and logs errors in the web service run-time log.

Input Transformation

The Input transformation represents the input element and header elements in the web service WSDL. The Input transformation receives the SOAP request from the client. It parses the XML message into groups of relational data and passes the data to other transformations in the operation mapping.

The Developer tool creates the Input transformation when you define the operation input for a web service.

Use the Input transformation **Ports** tab to view the operation input hierarchy, define output ports, and map data from the operation input to the output ports. The operation input hierarchy defines the SOAP request message hierarchy.

You can map the complete SOAP request as XML instead of returning groups of relational data in separate output ports. When you map the SOAP request as XML, the Data Integration Service returns the complete SOAP message in one output port.

Input Transformation Ports Tab

Define output groups, define output ports, and map nodes from the operation input to the output ports on **Ports** tab.

Choose to show the ports if you do not need to view the operation input hierarchy. When you show the ports, you can define groups, define ports, and map nodes from the operation input to the output ports. To map a node from the operation input to an output port, click the field in the **Location** column and expand the hierarchy in the **Select Location** dialog box. Then, choose a node from the hierarchy.

Choose to show the input mapping to view the operation input hierarchy. The left side of the tab is the **Operation Input** area and the right side of the tab is the **Ports** area. The **Operation Input** area shows the SOAP request message hierarchy. You can define the output ports in the **Ports** area. When you map a node from the operation input to an output port, the location of the node appears in the **Location** column in the **Operation Input** area.

When you show the input mapping, you can choose to display the output ports in a hierarchy. You can also choose to view the lines that connect the input ports to the nodes in the operation input.

The Developer tool maps nodes in the first level of the operation input to output ports when you choose to map the first level of the hierarchy. The Developer tool also creates the output ports that is requires to map the data. If the first level of the hierarchy contains a multi-occurring parent node with one or more multi-occurring child nodes, the Developer tool does not create the ports or map the first level of the hierarchy.

Rules and Guidelines to Map the Operation Input to Ports

When you configure the Input transformation, you map nodes from the operation input hierarchy to output ports.

Consider the following rules and guidelines when you map nodes from the operation input hierarchy to output ports:

- The node and the output port must have compatible datatypes.
- You cannot map a node to more than one output port in a group.

Configuring the Input Transformation

If the web service operation receives a SOAP request message, configure the Input transformation to process the request. Define output ports in the transformation. Map nodes from the operation input to the output ports.

1. Select the Input transformation in the editor.
2. Click the **Ports** tab of the **Properties** view.
3. Click **Input mapping**.

   The **Operation Input** area shows the request message hierarchy. Define the output ports in the **Ports** area.
4. Optionally, click **Show Lines** to view the lines that connect the output ports to the nodes in the operation input.

   You can choose to view all the lines or to view the lines for the selected ports.
5. Optionally, click **Show as Hierarchy** to display the output ports in a hierarchy.

   Each child group appears under the parent group.
6. If the operation input includes anyType elements, any elements, anyAttribute attributes, derived type elements, or substitution groups, choose objects in the **Operation Input** area. In the **Type** column for a node, click **Choose** and then choose one or more types, elements, or attributes from the list.
7. To add an output group, use one of the following methods:

Option	Description
Drag a node	Drag the pointer from a group node or a child node in the **Operation Input** area to an empty column in the **Ports** area. If the node is a group node, the Developer tool adds a group without ports. If other output groups exist, the **Map to new group** dialog box prompts you to relate the group to another group. The Developer tool creates keys for related groups.
Manually add a group	Click **New** > **Group** to add a group.
Select **Map first level hierarchy**	Select **Map first level hierarchy**. The Developer tool maps nodes in the first level of the operation input to output ports and groups. The Developer tool also creates the output ports and groups that it requires to map the data. If the first level of the hierarchy contains a multi-occurring parent node with one or more multi-occurring child nodes, the Developer tool does not create the ports or map the first level of the hierarchy.

8. To add output ports and map nodes to the output ports, use one of the following methods:

Option	Description
Drag a node	Drag the pointer from a node in the operation input to a group name or port in the **Ports** area.
Click the **Map** button	Select one or more nodes in the **Operation Input** area. Select a destination in the **Ports** area. Click **Map**.
Copy ports	Select ports from another transformation and copy them to the **Operation Input** area. To copy ports, you can use keyboard shortcuts or you can use the copy and the paste buttons in the Developer tool.

Option	Description
Manually add a port	Click **New** > **Field** to add a port.
Select **Map first level hierarchy**	Select **Map first level hierarchy**. The Developer tool maps nodes in the first level of the operation input to output ports and groups. The Developer tool also creates the output ports and groups that it requires to perform the mapping. If the first level of the hierarchy contains a multi-occurring parent node with one or more multi-occurring child nodes, the Developer tool does not create the ports or map the first level of the hierarchy.

9. To clear the locations of ports, use one of the following methods:

Option	Description
Click the **Clear** button	Select one or more ports in the **Ports** area and click **Clear**.
Delete the lines that connect nodes to ports	Select one or more lines that connect the nodes in the operation input to the output ports and press **Delete**.

10. To map the complete SOAP request as XML, right-click the **Request** node in the **Operation Input** area and select **Map as XML**.

Output Transformation

The Output transformation represents the output element and header elements in the web service WSDL. The Output transformation creates a SOAP response message from groups of relational data in the operation mapping. A WSDL might describe a large SOAP message hierarchy, but a web service operation might return data for part of the SOAP response message.

The Developer tool creates an Output transformation when you define the web service operation output.

Use the Output transformation **Ports** tab to view the operation output hierarchy, add transformation input ports, and map the input ports to the operation output. The operation output hierarchy defines the SOAP response message hierarchy.

You can map XML data from one string or text input port to the entire SOAP response. When you map XML data to the entire SOAP response, you cannot map ports to nodes in the operation output.

Configure advanced properties in the Output transformation **Advanced** tab.

Output Transformation Ports Tab

Define input groups, define input ports, and map input ports to operation output nodes on the **Ports** tab.

When you show the ports, you can manually add groups and ports or you can copy ports from other transformations into the Output transformation. You can use keyboard shortcuts or you can use the copy and the paste buttons in the Developer tool.

When you show the output mapping, you can define input groups, define input ports, and map input ports to operation output hierarchy. The left side of the tab is the **Ports** area and the right side of the tab is the **Operation Output** area. The **Operation Output** area shows the SOAP response message hierarchy. You can define input groups and input ports in the **Ports** area. When you map the input ports from the **Ports** area to

nodes in the **Operation Output** area, the input port location appears in the **Location** column in the Operation Output area.

The Developer tool maps input ports to nodes in the first level of the operation output when you choose to map the first level of the hierarchy. The Developer tool also creates the input ports that it requires to map the data. If the first level of the hierarchy contains a multi-occurring parent node with one or more multi-occurring child nodes, the Developer tool does not create the ports or map the first level of the hierarchy.

When you show the output mapping, you can choose to view the lines that connect the input ports to the nodes in the operation input.

Output Transformation Advanced Tab

Configure the Output transformation advanced properties on the **Advanced** tab.

The following table describes the property that you can configure on the Output transformation **Advanced** tab:

Property	Description
Sorted Input	Enables the Data Integration Service to generate output without processing all of the input data. Enable sorted input when the input data is sorted by the keys in the operation input hierarchy. Default is disabled.

Rules and Guidelines to Map Ports to the Operation Output

When you configure the Output transformation, you map ports to the operation output hierarchy.

Consider the following rules and guidelines when you map input ports to the operation output hierarchy:

- The input port and the node must have compatible datatypes.
- You can map an input port to one node in the hierarchy.
- You can map ports from one input group to nodes in the same hierarchy level in the operation output.
- You can map different ports from one input group to nodes in different hierarchy levels in the operation output.
- Map input ports to the keys in the operation output. Any port that you map to a key must be a string, integer, or bigint datatype. Map data to the keys in all levels in the operation output above the hierarchy level that you are including in the SOAP message. Include the foreign keys for all levels above and including the level you are mapping.

  Note: You do not have to map input ports to keys if you are mapping only the lowest level of the operation output hierarchy.
- You can map multiple input ports of different datatypes to a key. When you click the **Location** field for a key, you can reorder the input ports or remove one of the ports.

Configuring the Output Transformation

If the web service operation returns a response message, configure the Output transformation. Define input ports in each transformation and map data from the input ports to nodes in the operation output hierarchy.

1. Select the Output transformation in the editor.
2. Click the **Ports** tab of the **Properties** view.

3. Click **Output Mapping**.

   The **Operation Output** area shows the operation output hierarchy. Define the input ports in the **Ports** area.

4. Optionally, click **Show Lines** to view the lines that connect the input ports to the nodes in the operation fault. You can choose to view all the lines or to view the lines for the selected ports.

5. If the operation output includes anyType elements, any elements, anyAttribute attributes, derived type elements, or substitution groups, choose objects in the **Operation Output** area. In the **Type** column for a node, click **Choose** and then choose one or more types, elements, or attributes from the list.

6. To add an input group, use one of the following methods:

Option	Description
Drag a node	Drag the pointer from a group node or a child node in the **Operation output** area to an empty column in the **Ports** area. If the node is a group node, the Developer tool adds a group without ports.
Manually add a group	Click the arrow next to the **New** button and then click **New Group**.
Select **Map first level hierarchy**	Select **Map first level hierarchy**. The Developer tool maps nodes in the first level of the operation input to output ports and groups. The Developer tool also create the input ports and groups that it requires to map the data.

7. To add an input port, use one of the following methods:

Option	Description
Manually add a port	Click the arrow next to the **New** button and then click **New Port.**
Drag a port from another transformation	In the editor, drag a port from another transformation to the Output transformation.
Copy a port	Select ports from another transformation and copy them to the **Operation output** area. To copy ports, you can use keyboard shortcuts or you can use the copy and the paste buttons in the Developer tool.
Select **Map first level hierarchy**	Select **Map first level hierarchy**. The Developer tool maps nodes in the first level of the operation output to input ports and groups. The Developer tool also create the input ports and groups that it requires to perform the mapping.

8. To map data from the input ports to nodes in the operation output hierarchy, drag the pointer from each input port or group to the associated node in the Operation Output. The input field location appears next to the node in the **Operation Output** area.

9. To map XML data from an input port to the complete SOAP response, right-click the port and select **Map as XML**.

10. To map input ports as a composite key, use one of the following methods:

Option	Description
Drag input ports	Select two or more input ports and drag them to a key in the operation output hierarchy.

Option	Description
Select input ports from the **Select Location** dialog box	Click in the **Location** column of a key in the operation output hierarchy and then select the input ports.

11. To clear the locations of nodes, use one of the following methods:

Option	Description
Click the **Clear** button	Select one or more nodes in the **Operation Output** area and click **Clear**.
Delete the lines that connect ports to nodes	Select one or more lines that connect the input ports to the nodes in the operation output and press **Delete**.

Fault Transformation

The Fault transformation represents the Fault element in the web service WSDL. The Fault transformation generates a user-defined fault in the web service operation.

The Fault transformation represents either of the following user-defined faults:

Predefined fault

A predefined fault is where the Fault transformation represents the Fault element in the web service WSDL. In a predefined fault, the Fault transformation creates an error message from relational data in the web service operation mapping.

Generic fault

A generic fault is where the Fault transformation does not represent any of the Fault elements defined in the web service WSDL for a web service operation. The Fault transformation for a generic fault returns a generic error message when an error occurs in a transformation. You can create and configure a Fault transformation for a generic fault from the Developer tool.

An operation mapping can contain multiple Fault transformations. You can add multiple instances of the same Fault transformation in a mapping to generate the same message in different parts of the mapping.

You can add Fault transformations to the mapping or remove them from the mapping without changing the operation signature. If you know that a fault error can never occur when the mapping runs, you can remove the Fault transformation from the mapping. You must connect a Fault transformation to an upstream transformation or the mapping is not valid.

When you create a fault in an operation that has a SOAP 1.1 binding, the wizard creates the faultcode, faultstring, and faultactor elements. When you create a fault in an operation that has a SOAP 1.2 binding, the wizard creates the code, reason, node, and role elements. When you add elements to the fault, the wizard adds the elements to the detail group in the fault. Use the Fault transformation **Ports** tab to view the operation fault hierarchy, add transformation input ports, and map the input ports to the operation fault. The operation fault hierarchy defines the SOAP response message hierarchy for error messages that result from a user-defined fault.

You can map XML data from one string or text input port to the entire SOAP response. When you map XML data to the entire SOAP response, you cannot map ports to nodes in the operation fault.

Configure advanced properties in the Fault transformation **Advanced** tab.

Fault Transformation Ports Tab

Define the input groups, define input ports, and map input ports to operation fault nodes on the **Ports** tab.

When you show the ports, you can manually define groups and ports. Or, you can copy ports from other transformations into the Fault transformation. You can use keyboard shortcuts or you can use the copy and the paste buttons in the Developer tool.

When you show the fault mapping, you can define input groups, define input ports, and map input ports to operation fault hierarchy. The left side of the tab is the **Ports** area and the right side of the tab is the **Operation fault** area. The **Operation fault** area shows the SOAP response message hierarchy. When you map the input ports from the Ports area to nodes in the **Operation fault** area, the input port location appears in the **Location** column in the **Operation fault** area.

The Developer tool maps input ports to nodes in the first level of the operation fault when you choose to map the first level of the hierarchy. The Developer tool also creates the ports that it requires to map the data.

Fault Transformation Advanced Tab

Configure the Fault transformation advanced properties on the **Advanced** tab.

The following table describes the property that you can configure on the **Advanced** tab:

Property	Description
Sorted Input	Enables the Data Integration Service to generate output without having to process all of the input data. Enable sorted input when the input data is sorted by the keys in the operation fault hierarchy. Default is disabled.

Rules and Guidelines to Map Ports to the Operation Fault

When you configure the Fault transformation, you map input ports to the operation fault hierarchy.

Consider the following rules and guidelines when you map input ports to the operation fault hierarchy:

- You can map an input port to one node in the operation fault hierarchy. The input port and the node must have compatible datatypes.
- You can map ports from one input group to nodes in the same hierarchy level in the operation fault.
- You can map different ports from one input group to nodes in different hierarchy levels in the operation fault.
- You must map input port data to the keys in the operation fault hierarchy. Any port that you map to a key must be a string, integer or bigint datatype. Map data to the keys in all levels in the operation fault above the hierarchy level that you are including in the SOAP message. Include the foreign keys for all levels above and including the level you are mapping.
- You can map multiple input ports of different datatypes to a key. When you click the **Location** field for a key, you can reorder the input ports or remove one of the ports.

Creating a Fault Transformation

You can create a generic fault or a predefined Fault transformation. For a generic fault, the web service WSDL does not define the fault element. For a predefined fault, the web service uses a fault element to define the fault.

1. Open a web service.

2. In the **Outline** view, select the operation mapping.

   The operation mapping appears in the editor.

3. In the editor, right-click and select **Add Transformation**.

   The **Add Transformation** dialog box appears.

4. Select **Fault** and click **OK.**

   The **Add Fault** dialog box appears.

5. To create a fault transformation, complete one of the following steps.

   - Select **Create as a generic fault**.
   - Select **Create as a predefined fault from a fault element**.

     Note: The web service contains an element that defines the fault.

6. Click **OK.**

   The Fault transformation appears as a generic fault or a predefined fault.

Configuring the Fault Transformation

If the web service operation returns faults or if you create a Fault transformation for a generic fault, configure each Fault transformation. Define input ports and map data from the input ports to nodes in the operation fault.

1. Select the Fault transformation in the editor.
2. Click the **Ports** tab of the **Properties** view.
3. Click **Fault mapping**.

   The **Operation fault** area shows the response or fault message hierarchy. Define the input ports in the **Ports** area.

4. Optionally, click **Show Lines** to view the lines that connect the input ports to the nodes in the operation fault. You can choose to view all the lines or to view the lines for the selected ports.
5. If the operation fault includes anyType elements, any elements, anyAttribute attributes, derived type elements, or substitution groups, choose objects in the **Operation Fault** area. In the **Type** column for a node, click **Choose** and then choose one or more types, elements, or attributes from the list.
6. To add an input group, use one of the following methods:

Option	Description
Drag a node	Drag the pointer from a group node or a child node in the **Operation fault** area to an empty column in the **Ports** area. If the node is a group node, the Developer tool adds a group without ports.
Manually add a group	Click the arrow next to the **New** button and then click **New Group**.
Select **Map first level hierarchy**	Select **Map first level hierarchy**. The Developer tool maps nodes in the first level of the operation fault to input ports and groups. The Developer tool also create the input ports and groups that it requires to map the data.

7. To add an input port, use one of the following methods:

Option	Description
Manually add a port	Click the arrow next to the **New** button and click **New Port**.
Drag ports from other transformations	In the editor, drag a port from another transformation to the Fault transformation.
Copy ports	Select ports from another transformation and copy them to the **Operation Fault** area. To copy ports, you can use keyboard shortcuts or you can use the copy and the paste buttons in the Developer tool.
Select **Map first level hierarchy**	Select **Map first level hierarchy**. The Developer tool maps nodes in the first level of the operation fault to input ports and groups. The Developer tool also create the input ports and groups that it requires to map the data.

8. To map data from the input ports to nodes in the operation fault hierarchy, drag the pointer from each input port or group to the associated node in the operation fault.

   The input field location appears next to the node in the **Operation fault** area.

9. To map XML data from an input port to the complete SOAP response, right-click the port and select **Map as XML**.

10. To map input ports as a composite key, use one of the following methods:

Option	Description
Drag input ports	Select two or more input ports and drag them to a key in the operation fault hierarchy.
Select input ports from the **Select Location** dialog box	Click in the **Location** column of a key in the operation fault hierarchy and then select the input ports.

11. To clear the locations of nodes, use one of the following methods:

Option	Description
Click the **Clear** button	Select one or more nodes in the **Operation Fault** area and click **Clear**.
Delete the lines that connect ports to nodes	Select one or more lines that connect the input ports to the nodes in the operation fault and press **Delete**.

Fault Handling

When an error occurs in a web service, the Data Integration Service generates an error message and returns the message in a fault to the web service client.

If a SOAP 1.1 request is sent to an operation with SOAP 1.2 binding, the web service will generate a fault using SOAP 1.1. If a SOAP 1.2 request is sent to an operation with a SOAP 1.1 binding, the web service will generate a fault using SOAP 1.2. A web service can generate system defined faults and user-defined faults.

The Data Integration Service returns an error message to a web service client when a user-defined error occurs. A user-defined fault can be one of the following types:

- Predefined
- Generic

When the operation mapping contains an Output transformation, the web service returns data from the Output transformation or it returns a fault. If a fault occurs after the operation mapping has committed data to target transformations or external applications, the Data Integration Service cannot roll back the data. The mapping stops and the Data Integration Service discards the data that the web service Output transformation received.

SOAP 1.1 Fault

For SOAP 1.1, a fault is a SOAP message with the following structure:

```
Fault (FaultName)
Key_Fault (FaultName)
faultcode            xs:QName
faultstring          xs:string
faultactor           xs:anyURI
detail
```

The fault contains the following elements:

Fault code

A fault identification code such as an error message number.

Fault string

An explanation of the error.

Fault actor

Optional information about the object that caused the fault to occur.

Detail

Optional information that varies based on the fault.

SOAP 1.2 Fault

For SOAP 1.2, a fault is a SOAP message with the following structure:

```
Fault (FaultName)
Key_Fault (FaultName)
Code                tns:faultcodeEnum
Reason              tns:reasontext
Node                xs:anyURI
Role                xs:anyURI
detail
```

The fault contains the following elements:

Code

A fault identification.

The Value element of Code must be one of the following values:

- infasoapns:DataEncodingUnknown
- infasoapns:MustUnderstand

- infasoapns:Receiver
- infasoapns:Sender
- infasoapns:VersionMismatch

Note: You can expand the Code fault element to extract the SubCode fault element up to one level. The schema type for SubCode is xsd:QName. You can use the SubCode fault element to define an error message number.

Reason

An explanation of the error.

Node

Contains the URI of the SOAP node that generated the fault.

Role

Optional information about the object that caused the fault to occur.

Detail

Optional information that varies based on the fault.

System-Defined Faults

The Data Integration Service generates a system-defined fault when it encounters a system error. When a system error occurs, the Data Integration Service returns an error message in a system-defined fault.

For example, the Data Integration Service might return the following fault when a numeric input port receives data that is not numeric:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <soapenv:Fault>
         <faultcode>WSCMN_10022</faultcode>
         <faultstring>[WSCMN_10022] Mapping execution failed:
[com.informatica.platform.ldtm.common.ExecutionException: [MPSVCCMN_10009] The Mapping
Service Module [MappingService] encountered an exception with the following details:
[LDTM_0072] [ERROR] XML parsing component [Input_S2R] message code: [66022], message
body: A data conversion error occurred in field [customerID] of the transformation group
[Operation]. The field contains the following data: [1100AA].].</faultstring>
         <detail>         </detail>
      </soapenv:Fault>
   </soapenv:Body>
</soapenv:Envelope>
```

Note: The example uses SOAP 1.1.

You can return an error message to the web service client without a Fault transformation. You can raise a fault by calling the ABORT(msg) function in an Expression transformation. When you call an ABORT message, the Data Integration Service generates a system-defined fault message. The message that you pass into the ABORT function becomes the fault string in the system-defined fault.

Predefined Faults

For a predefined fault, the web service uses a fault element to define the fault. Configure a Fault transformation to return a custom error message.

When you configure a Fault transformation in a web service, you must define the operation mapping logic that returns the error condition. You need to define a transformation to generate the error message to pass to the Fault transformation.

When you define a Fault transformation, you define the data to return in the fault code, fault string, and fault actor. You can also add more elements in the fault message in order to return more information to the web service client. When you define the elements in the fault, the wizard adds the elements to the detail group. You can define multiple-occurring elements and create hierarchical relationships between groups of elements in the detail group.

You can use the same Fault transformation multiple times in a mapping when the error message structure is the same for each error. Otherwise, you can configure a different Fault transformation for each error message that you want to return to a web service client.

For example, you configure a Fault transformation to return messages to the web service client when an employee is not found in a lookup. You pass an error number to the faultcode element and an error message to the faultstring element. You also need to return the department ID and the employee ID to the web service client. When you define the fault, you add the DeptID and EmployeeID elements in the detail group.

The Data Integration Service might return the following fault:

```
<infasoapns:Envelope xmlns:infasoapns="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:infawsdlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://
www.informatica.com/dis/ws/Get_Employee_Info_Web_Service">
    <infasoapns:Body>
        <infasoapns:Fault>
            <faultcode>ERR_12345</faultcode>
            <faultstring>Web service failed to retrieve employee information.</
faultstring>
            <detail>
                <tns:Employee_Not_Found>
                    <tns:DeptID>100</tns:DeptID>
                    <tns:EmployeeID>2428/tns:EmployeeID>
                </tns:Employee_Not_Found>
            </detail>
        </infasoapns:Fault>
    </infasoapns:Body>
</infasoapns:Envelope>
```

Note: The example uses SOAP 1.1.

Generic Faults

You can define a generic fault to return an error message to a web service client when an error is not defined by a fault element in the WSDL. Create a Fault transformation to return a generic error message when an error occurs in a transformation.

When you create a Fault transformation for a generic fault in a web service, you must define the operation mapping logic that returns the error condition.

When you define a Fault transformation for a generic fault, and if the operation binding is in SOAP 1.1 format, you define the data to return in the fault code, fault string, and fault actor elements. If the operation binding is in SOAP 1.2 format, you define the data to return in the code, reason, node, and role elements. You can also optionally define data to return in the detail string. The detail string is optional and has an xsd:any element. If you want to send data to the detail string, map the detail element to one of the ports defined in the web service operation. You can also add more elements in the fault message to return more information to the web service client.

You can create hierarchical relationships between groups of elements in the detail group. You can use the same Fault transformation multiple times in a mapping when the error message structure is the same for each error. You can also create a different Fault transformation for each error message that you want to return to a web service client.

Note: After creating a fault transformation, you cannot change the status of a generic fault or a predefined fault.

Test Operation Mappings

Preview the output of an operation mapping to verify that it produces the results that you want. You can also preview the output of a transformation in the mapping.

Create a web service configuration to control the settings that the Developer tool applies when you preview the output of an operation mapping or the output of a transformation in the operation mapping. Use the **Preferences** dialog box to configure the default web service run configuration. You can also use the **Run** dialog box to create web service configurations that you can specify in the **Data Viewer** view.

Testing Operation Mappings

Test an operation mapping to preview the SOAP response for an operation. You can preview the output of an operation mapping or you can preview output from a transformation in the mapping.

Before you can preview data, you need to select a default Data Integration Service.

1. Open a web service.
2. Select the operation mapping in the **Outline** view.

   The operation mapping appears in the editor.
3. Select the **Data Viewer** view.
4. In the **Input** window, enter a request.
5. If you want to view the output data of a specific transformation, select the transformation in the editor. Otherwise, the output of the operation mapping displays when you run the data viewer.
6. Click **Run**.

   The result of the request appears in the **Output** window.
7. To test the operation mapping with a different request, click **Reset** and then repeat steps 4 through 6.

Customized View Options

When you configure the Input, Output, and Fault transformations, you can change the SOAP message hierarchy to show keys in the **Operation** area. You can also show grouping constructs that define how to order nodes.

To view the customized view options, click **Customize View** in the **Operation Input** area, the **Operation Output** area, or the **Operation Fault** area.

You can enable the following options:

All, Sequence, and Choice

Show a line that indicates whether an element definition is all, sequence, or choice.

Nodes in an all group must all be included in the SOAP message.

Nodes in a sequence group must be in the order specified in the WSDL.

At least one node in a choice group must appear in the SOAP message.

Keys

View the keys for each hierarchy level.

CHAPTER 6

Parsing Web Service SOAP Messages

This chapter includes the following topics:

- Parsing Web Service SOAP Message Overview, 60
- Transformation User Interface, 61
- Multiple-Occurring Output Configuration, 62
- Parsing anyType Elements, 64
- Parsing Derived Types, 65
- Parsing QName Elements, 66
- Parsing Substitution Groups, 66
- Parsing XML Constructs in SOAP Messages, 66

Parsing Web Service SOAP Message Overview

The Data Integration Service generates row data when it parses a SOAP message in a web service transformation.

The web service Input transformation and the Web Service Consumer transformation are web service transformations that parse SOAP messages.

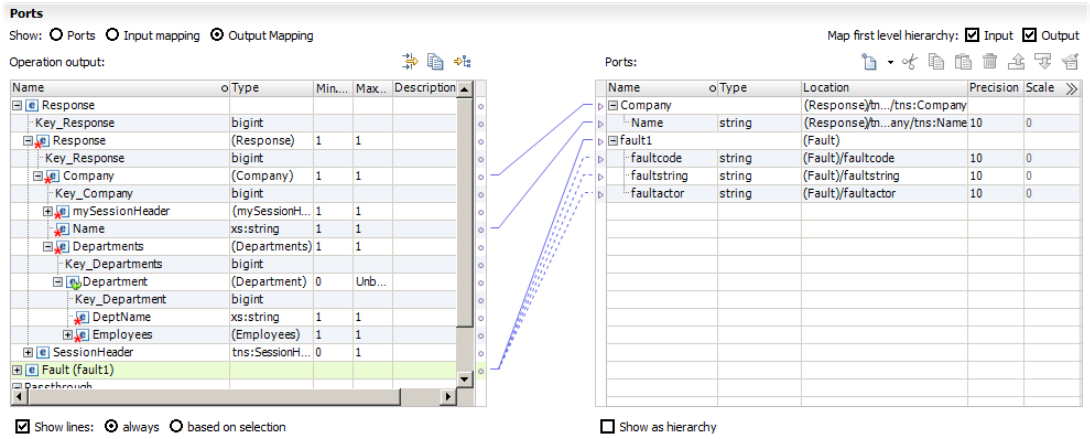
To configure a transformation to parse a SOAP message, create output ports in a structure similar to the SOAP message hierarchy. Map the nodes in the SOAP message hierarchy to the ports.

You can configure normalized groups of output ports, denormalized groups, and pivoted groups of ports. When the SOAP message contains derived types, anyType elements, or substitution groups, you can configure different output groups based on what types might occur in the SOAP messsage instance.

Transformation User Interface

The Web Service Consumer transformation and the web services Input transformation provide a user interface that you can use to map data from the SOAP message to the transformation output ports.

The following figure shows a mapping between SOAP 1.1 message nodes and output ports in a Web Service Consumer transformation:



Operation Area

The Operation area contains the SOAP message hierarchy. Complex nodes or multiple-occurring nodes define hierarchy levels in the structure. The Developer tool adds keys to the levels that define the parent-child relationships between them.

In the preceding figure, the SOAP message hierarchy has the following levels:

Response or Request

Level that represents the root of the response or request message.

Company

Top level of the request data.

Departments

Multiple-occurring departments within the company.

Employees

Employee is a complex element within a department.

Fault Group

Fault message group that receives error messages.

Ports Area

You can map data from levels of the SOAP message to output ports. Each group of output ports can be related to other output groups with primary foreign-key relationships.

In the preceding figure, the transformation has groups of output ports that correspond to the groups of nodes in the SOAP message.

Multiple-Occurring Output Configuration

When an Input transformation or Web Service Consumer transformation returns multiple-occurring data, you can configure the output ports in different configurations.

You can configure normalized output data, pivoted output data, or denormalized output data.

For example, a SOAP message contains Departments and Employees complex elements. Each department contains multiple employees. Departments is the parent of Employees.

The SOAP message contains the following element hierarchy:

```
Departments
      Department_ID
      Department_Name
      Employees
         Employee_ID
         Employee_Name
```

Normalized Relational Output

When you create normalized output data, the data values do not repeat in an output group. You create a one-to-one relationship between the hierarchy levels in the SOAP message and the output groups of ports.

When the SOAP message contains a Departments parent hierarchy level and an Employees child hierarchy level, you might create the following groups of ports:

```
Departments
    Department_Key
    Department_ID
    Department_Name


Employees
    Department_Key
    Employee_ID
    Employee_Name
```

The Department_Key is a generated key that relates the Employees output group to a Department group.

Generated Keys

When you add an output group, the Developer tool relates the output group to another output group with a generated key. The Developer tool adds a bigint key to the parent group and to the child group. At run time, the Data Integration Service creates the key values for the generated keys.

Example

The SOAP hierarchy has the following nodes:

```
Departments
   Dept_Key
   Dept_Num
   Dept_Name

   Employees
      Dept_FK
      Employee_Num
      Employee_Name
```

When you create an output group of ports for Departments, you map the Departments node to an empty field in the Ports area. The Developer tool creates the following output group:

```
Departments
   Dept_Num
   Dept_Name
```

When you map the Employees node to an empty field in the Ports area, the Developer tool prompts you to relate the Employees group to the Departments group. You can relate the Employees group to more than one group. The Developer tool adds a key to the each group.

The Developer tool creates the following groups and generated keys:

```
Departments
   Key_Departments
   Dept_Num
   Dept_Name

   Employees
      Key_Departments
      Employee_Num
      Employee_Name
```

Note: You do not have to map nodes to the generated keys. The Data Integration Service creates the key values at run time.

The Developer tool can create generated keys to multiple levels in one output group. The Employees group might contain the following ports:

```
Employees
   Key_Employees
   Key_Departments
   Key_Managers
   Employee_Num
   Employee_Name
```

Key_Departments and Key_Managers are the generated keys that point to parent groups. Key_ Employees is a generated key for the Employees group. Key_Employees appears when you relate a child group to the Employees group.

Denormalized Relational Output

You can denormalize relational output. When you denormalize the output data, the element values from the parent group repeat for each child element.

To denormalize output data, map nodes from the parent hierarchy level to the child group of output ports.

The following example shows the Department_ID and the Department_Name in the Employees output group:

```
Employees
    Department_ID
    Department_Name
    Employee_ID
    Employee_Name
```

Department_ID and Department_Name repeat for each employee in the department:

Department_ID	Department_Name	Employee_ID	Employee_Name
100	Accounting	56500	Kathy Jones
100	Accounting	56501	Tom Lyons
100	Accounting	56509	Bob Smith

Pivoted Relational Output

You can include a specific number of multiple-occurring elements in an output group.

To pivot multiple-occurring elements, map the multiple-occurring child element to the parent group of output ports. The Developer tool prompts you to define the number of child elements to include in the parent.

The following example shows two instances of Employee_ID in the Departments parent group:

```
Departments
    Department_ID
    Department_Name
    Employee_ID1
    Employee_ID2
```

Parsing anyType Elements

The anyType element represents a choice of all global types in a WSDL or schema. When you map nodes to ports in the Developer tool, you choose which types to appear in the SOAP message for the anyType element. You must replace an anyType element in the SOAP message with a complex type or xs:string. Create groups of ports for each type that you choose.

You must choose a type to map data to output ports. If the WSDL or schema does not contain a global type, the Developer tool replaces the anyType element with xs:string.

To choose an element type in the Operation area, click **Choose** in the **Type** column for the anyType element. A list of available complex types and xs:string appears.

When you replace an anyType element with derived types, the Data Integration Service populates elements for one type at a time. The SOAP message does not contain data for the base type and the derived type at the same time.

Derived Types Example

The WSDL contains an anyType element. You replace the element with AddressType and a derived type called USAddressType. The SOAP message hierarchy has the following groups:

```
Address:AddressType (base type)
  Address: AddressType
    Street
    City

  Address:USAddressType   (derived type)
     Street
     City
     State
     ZipCode
```

The SOAP message contains the following data:

```
<address xsi: type ="AddressType">
<street>1002 Mission St.</street>
<city>san jose</city>
</address>

<address xsi:type="USAddressType">
<street>234 Fremont Blvd</street>
<city>Fremont</city>
<zip>94556</zip>
<state>CA</state>
```

```
</address>
```

The Data Integration Service returns one row for xsi: AddressType:

Street	City
1002 Mission St.	San Jose

The Data Integration Service returns one row for the derived type xsi: USAddressType:

Street	City	State	Zip
234 Fremont Blvd.	Sunnyvale	CA	94556

The Data Integration Service does not populate the AddressType if the type is xsi: USAddressType.

Parsing Derived Types

You can parse SOAP messages that contain derived types. When you define the ports that receive the data from the SOAP message, choose which types might appear in a SOAP message. The elements in the types you choose determine the ports that you need to create.

For example, the WSDL might contain an AddressType and a derived type called USAddressType. You can create the following groups in the Developer tool Operation area:

```
Address
  Address: AddressType
    Street
    City

  Address:USAddressType
     Street
     City
     State
     ZipCode
```

The SOAP message might contain the following data:

```
<address>
<street>1002 Mission St.</street>
<city>san jose</city>
</address>

<address xsi:type="USAddressType">
<street>234 Fremont Blvd</street>
<city>Fremont</city>
<zip>94556</zip>
<state>CA</state>
</address>

<address xsi:type="USAddressType">
<street>100 Cardinal Way</street>
<city>Redwood City</city>
<zip>94536</zip>
<state>CA</state>
</address>

<address>
<street>100 El Camino Real</street>
<city>Sunnyvale</city>
</address>
```

The Data Integration Service returns the following rows for the base type, Address:

Street	City
1002 Mission St.	San Jose
234 Fremont Blvd	Sunnyvale
100 Cardinal Way	Redwood City
100 El Camino Real	Sunnyvale

The Data Integration Service returns the following rows for the derived type, USAddress:

Street	City	State	Zip
234 Fremont Blvd.	Sunnyvale	CA	94556
100 Cardinal Way	Redwood City	CA	94536

The Data Integration Service returns all addresses in the base type. The Data Integration Service returns US Addresses in the derived type. The derived type includes the Street and City elements that the USAddressType inhertits from the base type.

Parsing QName Elements

When Data Integration Service parses QName elements in the SOAP message, it updates QName values that belong to the namespace of the schema to use the namespace prefix defined in the schema. Otherwise, Data Integration Service does not update the value of the element.

For example, the schema has the namespace prefix `tns` defined for namespace `"http://user/test"`. The SOAP message has namespace prefix `mytns` defined for the same namespace. When the Data Integration Service parses Qname value `mytns:myelement` it changes the value to `tns:myElement`.

When Data Integration Service generates QName elements in the SOAP message, it does not update the value of the element.

Parsing Substitution Groups

A substitution group replaces one element with another element from the same group. Substitution groups are similar to derived types, except that each element definition includes a substitution group name.

You can configure an output group of ports that receives elements from a specific type in a substitution group. You can create a different output group of ports that receives an element from another type in the substitution group.

Parsing XML Constructs in SOAP Messages

A SOAP message might contain XML constructs such as choice, list, and union elements.

Web service transformations can parse SOAP messages with these constructs with some limitations.

Choice Element

A choice element restricts a child element to one of the elements in the <choice> declaration.

The following text shows a person element that is an employee or a contractor:

```
<xs:element name="person">
  <xs:complexType>
    <xs:choice>
      <xs:element name="employee" type="employee"/>
      <xs:element name="contractor" type="contractor"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

You can map choice elements using the following methods:

- Create output ports for each choice element in an output group. Some elements will have null values in the output row.
- Create an output group for each choice. For the above example, create an employee group and a contractor group. The Data Integration Service generates a row based on which element appears in the SOAP message.

List Element

A list is an XML element that can contain multiple simple type values, such as "Monday Tuesday Wednesday".

The Data Integration Service can return a list as a string value. When the SOAP message contains a list, you cannot map items from the list to separate output rows. You can configure an Expression transformation to separate the elements in the list if you need them separated in a mapping.

Union Element

The union element is a simple type that is a union of more than one type.

The following text shows an element Size that is a union of two simple types, size_no and size_string:

```
<xs:element name="Size">
  <xs:simpleType>
    <xs:union memberTypes="size_no size_string" />
  </xs:simpleType>
</xs:element>
```

To map the size to an output port, create one port for the size. Configure the output port as a string. You can configure another transformation in the mapping to convert the data to another type.

CHAPTER 7

Generating Web Service SOAP Messages

This chapter includes the following topics:

- Generating Web Service SOAP Messages Overview, 68
- Transformation User Interface, 69
- Port and Hierarchy Level Relationships , 70
- Keys, 71
- Map Ports, 72
- Pivoting Multiple-Occurring Ports , 74
- Map Denormalized Data, 75
- Derived Types and Element Substitution, 76
- Generating XML Constructs in SOAP Messages, 78

Generating Web Service SOAP Messages Overview

The Data Integration Service generates XML data from groups of input data when it generates a SOAP message. When you create a Web Service Consumer transformation, a web service Output transformation, or a Fault transformation, you configure which input ports to map to the SOAP message hierarchy.

To configure a transformation to generate a SOAP message, create groups of input ports and map each group to a group in the SOAP message hierarchy. A WSDL or schema defines the structure of the SOAP message.

You can configure groups of data in the SOAP message from denormalized input data. You can also pivot multiple-occurring input data to multiple-occurring nodes in the SOAP message.

You can map data to derived types, anyType elements, or substitution groups in a SOAP message. You must choose which types can occur in the SOAP message when you define a transformation. The types that you choose determine the input ports that you need to create.

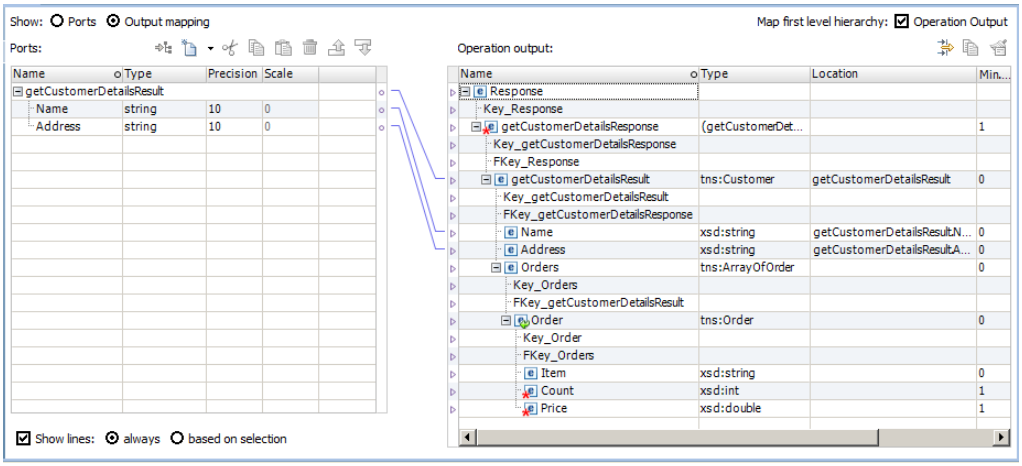
When you view the SOAP message hierarchy in the Developer tool, the hierarchy contains keys. The keys do not appear in the SOAP message. The Data Integration Service uses keys to define parent-child relationships between groups in the SOAP message. To configure key values, map input data to keys in the SOAP message.

Transformation User Interface

The web services Output transformation, the Fault transformation, and the Web Service Consumer transformation contain a user interface that you use to configure the SOAP message.

To configure a transformation to generate a SOAP message, create input ports in a structure similar to the SOAP message hierarchy. The WSDL or schema determines the structure of the hierarchy. Map each input port to a node in the SOAP message.

The following figure shows a mapping between the input ports and the SOAP message nodes in a web service Output transformation:



Input Ports Area

Create groups of input ports in the **Input Ports** area. Include input ports for each level in the SOAP message hierarchy that you need to map.

You must create a Response or Request input group and the child groups that receive the data.

When you create input port groups, define a primary key in each parent group. Define a foreign key in each child group. The foreign key relates the group to a parent group.

You do not have to define keys for the Response level or the WSDL root level unless you are passing data at the WSDL root level. For example, the root level might contain HTTP headers.

You might create groups of ports similar to the following groups for customers and orders:

```
Response
   Response_Key

   Customer_Details_Root
      Key_Cust_Det
      FK_Response_Key

      Customer
        Customer_ID
        FK_Cust_Det
        Name
        Address

          Orders
            Order_Num
            FK_Cust_ID
```

```
Order_Items
  Order_Num
  Item
  Count
  Price
```

Operation Area

The **Operation** area shows the elements in the SOAP message hierarchy as defined by the WSDL or schema. The SOAP message does not have to contain all the elements from the WSDL or schema. The message contains the data that you map from the input ports.

Multiple-occurring nodes and complex nodes define hierarchy levels in the SOAP message structure. The Developer tool adds keys to the levels to create parent-child relationships between them. All levels in the hierarchy, except for the leaf levels, have a primary key. Each child level has a foreign key to a parent level. The keys that appear in the SOAP message hierarchy do not appear in a SOAP message instance. The Data Integration Service requires values in the keys to relate levels of the data when it generates the SOAP message.

The **Location** column contains the group name and input port that contains the data for an element in the SOAP message. The **Location** column is empty until you map an input port to the node.

In the preceding figure, the SOAP message contains a single instance of customer details and orders. The Orders group contains a multiple-occurring element called Order. The SOAP message hierarchy has the following levels related by key:

```
Response
   GetCustomerDetailsResponse
      GetCustomerDetailsResult
         Orders
            Order
```

The Response level represents the root of the response message. The Data Integration Service requires this level to attach headers to the SOAP message.

The GetCustomerDetailsResponse level is the root of the message.

Port and Hierarchy Level Relationships

When you map input ports to the SOAP message hierarchy, you maintain a relationship between an input group and a SOAP message hierarchy level. For example, you might have two input groups, Department and Employee.

The Department input group receives the following rows:

Dept_num	Name	Location
101	HR	New York
102	Product	California

The Employee input group receives the following rows:

Dept_num	Employee
101	Alice
101	Bob
102	Carol
102	Dave

Map the department number in the Employee group as a foreign key that establishes the relationship between the Department and the Employee group. The department number occurs in the department hierarchy level, but not the employee level.

The SOAP message contains the following XML structure:

```
<department>
   <dept_num>101</dept_num>
   <name>HR</name>
   <location>New York</location>

   <employee>
      <name>Alice</name>
   </employee>

   <employee>
      <name>Bob</name>
   </employee>
</department>

<department>
   <dept_num>102</dept_num>
   <name>Product</name>
   <location>California</location>

   <employee>
      <name>Carol</name>
   </employee>

   <employee>
      <name>Dave</name>
   </employee>

</department>
```

Keys

A SOAP message hierarchy includes keys. The Data Integration Service requires key values to construct the XML hierarchy in the SOAP message.

You must map input port data to the keys in the SOAP message hierarchy. Map data to the keys in each level that you are providing data. When you have a multiple-occurring node, you need to relate the node to a parent.

The keys appear in the SOAP message without types. Any port that you map to a key must be a string, integer, or bigint datatype. The primary key in the parent group and the foreign key in each child group must have the same datatype, precision, and scale. You can map generated keys to the SOAP message keys.

You can map a port to a node and to a key in the same hierarchy level. For example, you map Employee_ID to a node in the SOAP message and you map it to a key in the Employee level.

If two group nodes in the hierarchy have a parent-child relationship, complete the following tasks:

- Map a port to the primary key in the parent node group.
- Map a port to the foreign key in the child node group.

You can also map primary keys to input ports to remove rows with a primary key of null or with duplicate primary keys.

You can create a composite key in a SOAP message by mapping multiple ports to the same key. Use composite keys when you need to denormalize data and maintain unique keys for some multiple-occurring combinations of values. You can create composite keys that contain strings, bigint, or integer values.

Note: You can include an Expression transformation in the operation mapping to generate key values.

Composite Key Example

Configure a unique division-department key from the following groups of ports:

```
Company
   Company_Num
   Company_Name

   Division
      Company_Num
      Divison_Num
      Division_Name

      Department
         Division_Num
         Dept_Num
         Dept_Name
         Location
```

The Dept_Num is unique within a division, but Dept_Num is not unique for all divisions in the Company.

You might configure a Department group that contains the division and the department information. Configure the division number and the department number as part of the composite key:

```
      Department
         Division_Num + Dept_Num (key)
         Dept_Name
         Location
```

The order that you map the ports determines the key value.

Map Ports

After you create input ports, map each input port to the SOAP message hierarchy. The location of the port appears next to the node in the **Operation** area.

You can map ports to the following types of nodes:

Atomic node

A simple element or an attribute that has no children and is not divisible.

Multiple-occurring atomic node

A simple element or an attribute that occurs multiple times at the same location in the hierarchy.

Complex node

An element that contains other elements.

If the parent node does not have a location, the parent node receives the input group name as the location. When the parent node has a location, each node in the hierarchy level must have an output location from the same location.

You can map an input group name to a parent node in a hierarchy level. The Developer tool updates the location field for the parent node in the hierarchy. The Developer tool does not update the child nodes that belong to the group in the hierarchy. When you map input ports to the child nodes, each input port location must be the same location as the location parent node.

After you map an input group to a hierarchy level, you can change it later. You can click **Clear** or you can delete the lines between the Ports and the Operation area. To delete the lines, drag the pointer of the lines to select them. Click **Delete**.

Map a Port

When you map a port to a node in the SOAP message, the Developer tool provides different results based on the type of node to which you map the port.

The following table describes the results when you map a single port to different target nodes in the **Operation** area:

Target Node	Results
Atomic node	When you map a single port to a node and the parent node does not have a location, the node receives the location of the port. The parent node location receives the location of the input group for the single port. When you map a single port to a node and the parent node already has a location, you can change the location for the parent node and clear the location for the other child nodes in the same level. The hierarchy level location changes to the group name of the port.
Multiple-occurring atomic node or the primary key of the multiple-occurring atomic node	When you map a single port to the multiple-occurring atomic node, the Developer tool sets the location for the atomic node to the group of the selected port.
Complex node	When you map a single port to a complex node, the Developer tool sets the location of the complex node to the location of the group that contains the port. The Developer tool prompts you for the single occurring atomic node to assign the port to. If all single-occurring atomic nodes have a location, you cannot map the complex node.

Map a Group

When you map an input group to a node in the SOAP message, the Developer tool provides different results based on the type of node that you map the port to.

The following table describes the results when you map a group to a node in the **Operation** area:

Target Node	Results
Atomic node	You cannot map a group to an atomic node.
Multiple-occurring atomic node	You are prompted to choose a port in the input group to update the location for the node and primary key.
Multiple-occurring complex node	The Developer tool sets the location for the complex node to the location of the group.

Map Multiple Ports

When you map multiple ports to a node in the SOAP message, the Developer tool provides different results based on the type of node you map the ports to. You can map multiple ports at the same time if you map them from the same group.

The following table describes the results for the node when you map multiple ports to nodes:

Target Node	Results
Single atomic node	When you map multiple ports to a single node, you update the location for more than one single atomic node in the **Operation** area. If the hierarchy does not have enough nodes in the level to update, the Developer tool maps ports just for the available nodes.
Multiple-occurring atomic node	When you map multiple ports to multiple-occurring atomic node, you pivot the ports into multiple occurrences of the node. The Developer tool creates instances of the node based on how many ports you map. A message appears that describes the number of ports that you projected.
Multiple-occurring complex node	When you map multiple ports to a complex node, you must select which single-occurring nodes atomic nodes to update. You pivot the ports into multiple occurrences of the node. The Developer tool creates instances of the node based on how many ports you map.

Pivoting Multiple-Occurring Ports

You can map multiple input ports to a multiple-occurring node in the SOAP message. The Developer tool pivots the input data into multiple nodes in the SOAP message.

To change the number of elements to pivot, choose **Override existing pivoting** in the **Map Options** dialog box.

If you remove one of the pivoted port instances from the **Ports** area, the Developer tool removes all instances from the **Operation** area.

Pivoting Example

An input group might have the following rows:

Num	Name	Location	emp_name1	emp_name2	emp_name3
101	HR	New York	Alice	Tom	Bob
102	Product	California	Carol	TIm	Dave

Each row contains a department number and three employees names.

Employee is a multiple-occurring node in the SOAP message hierarchy. You can map all instances of Employee from the input row to the SOAP message hierarchy. Select all occurrences of Employee. Click **Map**. The **Map Options** dialog box prompts you to choose a node from the list.

The Developer tool changes the Employee node to include the multiple name nodes in the SOAP message hierarchy:

```
Department
   num
   name
   location
   Employee     (unbounded)
       emp_name1
       emp_name2
       emp_name3
```

The SOAP message returns the following hierarchy:

```
<department>
   <num>101</num>
   <name>HR</name>
   <location>New York</location>
   <employee>
      <emp_name>Alice</name>
   </employee>
   <employee>
      <emp_name>Tom</name>
   </employee>
   <employee>
      <emp_name>Bob</name>
   </employee>
</department>

<department>
   <num>102</num>
   <name>Product</name>
   <location>California</location>
   <employee>
      <emp_name>Carol</name>
   </employee>
   <employee>
      <emp_name>Tim</name>
   </employee>
   <employee>
      <emp_name>Dave</name>
   </employee>
  </department>
```

Map Denormalized Data

You can map denormalized data and pass it to normalized nodes in a SOAP message.

When you map denormalized data, you pass data from one input group to multiple nodes in the SOAP message hierarchy. You can create group relationships in the SOAP message similar to the following types of relationships:

Linear Node Relationship

Node A is parent to Node B. Node B is a parent to Node C. Node C is the parent of Node D.

Hierarchical Node Relationship

Node A is a parent to Node B. Node A is also a parent to Node C. Node B and Node C are not related.

The following table shows input rows that contain denormalized division and department data:

Division	Dept_Num	Dept_Name	Phone	Employee_Num	Employee_Name
01	100	Accounting	3580	2110	Amir
01	100	Accounting	3580	2113	Robert
01	101	Engineering	3582	2114	Stan
01	101	Engineering	3582	2115	Jim
02	102	Facilities	3583	2116	Jose

The input data contains unique employee numbers and names. The division and department data repeat for each employee in the same department and division.

Linear Group Relationship

When you configure ports, you can configure a separate group for Division, Department, and Employee. Division is a parent of Department, and Department is the parent of Employee. You can configure groups in the following linear structure:

```
Division
   Division_Key
   Division_Num
   Division_Name

      Department
        Department_Key
        Division_FKey
        Dept_Num
        Dept_Name
        Phone

        Employee
           Department_Fkey
           Employee_Num
           Employee_Name
```

The SOAP message contains unique instances of Division and Department although Division_Num and Dept_Num repeat in the input data. Define the Division_Num as the primary key in the Division group. Define Dept_Num as the primary key in the Department group.

Hierarchical Group Relationship

You can create a group hierarchy that contains the Division parent group and Department and Employee child groups. Department and Employee do not have a primary key-foreign key relationship. Department and Employee are children of Division. You can configure the groups in the following structure:

```
Division
   Division_Key
   Division_Num
   Division_Name

   Department
      Division_FKey
      Dept_Num
      Dept_Name

   Employee
      Division_FKey
      Employee_Num
      Employee_Name
```

Derived Types and Element Substitution

You can map input ports to derived complex types, anyType elements, and substitution groups in a SOAP message. The SOAP message can include elements for the base type and the derived types.

In a type relationship, the base type is the type from which you derive another type. A derived type inherits elements from the base type. An extended complex type is a derived type that inherits elements from a base type and includes additional elements. A restricted complex type is a derived type that restricts some elements from the base type.

Generating Derived Types

When a WSDL or schema includes derived types, you must choose the types that you want to include in the SOAP message.

For example, the WSDL defines a base type AddressType. The WSDL also contains a USAddressType and a UKAddressType that are the derived AddressTypes.

Each type contains the following elements:

- AddressType: street, city
- USAddressType (extends AddressType): state, zipCode
- UKAddressType (extends AddressType): postalCode, country

When you choose a USAddressType in the Operation area, the Developer tool creates a group for the USAddressType element in the SOAP message. The group includes the street and city from the base address and the state and zipCode for the USAddress. Derived types that extend base types always include the elements from the base type.

If you choose all of the available derived types for the SOAP messsage, the Developer tool creates groups similar to the following in the SOAP hierarchy:

```
Address
  Address: Address
    Street
    City

  Address:USAddressType
     Street
     City
     State
     ZipCode

  Address: UKAddressType
     Street
     City
     PostalCode
     Country
```

You need to define input port groups for Address, USAddress, and UKAddress.

Generating anyType Elements and Attributes

Some schema elements and attributes allow any type of data in a SOAP message.

The anytype element represents a choice of all globally known types. Before you map a port to an anyType element in a SOAP message, choose an available complex type or xs:string. If the WSDL or schema does not contain a complex type, the Developer tool replaces the anyType element type with xs:string.

To choose an element type in the Operation area, click **Choose** in the **Type** column for the anyType element. A list of available complex types and xs:string appears.

The following element and attributes allow any type of data:

anyType element

Allows an element to be any datatype in the associated XML file.

anySimpleType element

Allows an element to be any simpleType in the associated XML file.

ANY content element

Allows an element to be any global element defined in the schema.

anyAttribute attribute

Allows an element to be any attribute already defined in the schema.

Generating Substitution Groups

Use substitution groups to replace one element with another in a SOAP message. Substitution groups work similarly to derived types, except that the element definitions include a substitution group name.

For example, you might have a base type Address and the derived types USAddress and UKAddress:

```
xs:element name="Address" type="xs:string"/>
<xs:element name="USAddress" substitutionGroup="Address"/>
<xs:element name="UKAddress" substitutionGroup="Address"/>
```

When you configure the SOAP message hierarchy, you can choose which element to substitute for Address in the SOAP message.

Generating XML Constructs in SOAP Messages

A WSDL or schema can contain choice, list, or union elements. Web service transformations can generate SOAP messages that contain these elements.

Choice Element

A choice element restricts a child element to one of the elements in the <choice> declaration.

To map ports to a SOAP message that contains choice elements, create one input group that includes all the elements in the choice construct. For example, an item description might be a dimension or a weight:

```
item: description, choice {dimension, weight}
```

When the description is a dimension, the description is a complex type that contains, length, width, and height.

When the description is a weight, the description is a simple string type.

The input data has the following columns and rows:

description	length	width	height	weight
box	20cm	18cm	15cm	NULL
coffee	NULL	NULL	NULL	500g

The SOAP message contains an Item group that contains dimensions or weight descriptions:

```
Item
   Description
       Dimension
          Length
          Width
          Height
       Weight
```

The NULL values in the input data become missing elements in the XML output.

The SOAP message contains the following data:

```
<item>
   <desc>box</desc>
      <dimension>
```

```
            <length>20cm</length>
            <width>18cm</width>
            <height>15cm</height>
          </dimension>
    </item>

    <item>
       <desc>coffee</desc>
          <weight>500g</weight>
    </item>
```

List Element

A list is an XML element that can contain multiple simple type values in the same element or attribute. The Data Integration Service can process a list in the input data if the list is represented as consolidated string of data.

If each item in the list is a separate element, such as ClassDates1, ClassDates2, and ClassDates3, the Data Integration Service cannot process the items as a list. You can use an Expression transformation to combine them into a string if you need to return a list in a SOAP message.

The following input rows contain a list element called ClassDates that contains days of the week:

CourseID	Name	ClassDates
Math 1	Beginning Algebra	Mon Wed Fri
History 1	World History	Tue Thu

The Data Integration Service can return a SOAP message with the following XML structure:

```
<class>
   <courseId>Math 1</courseId>
   <name>Beginning Algebra</name>
   <classDates>Mon Wed Fri</classDates>
</class>
<class>
   <courseId>History 1</courseId>
   <name>World History</name>
   <classDates>Tue Thu</classDates>
</class>
```

Union Element

The union element is a simple type that is a union of more than one type. When a SOAP message contains a union element, you must map a single input port that contains the data in a string.

For example, the SOAP message contains an element called size. Size is a union of integer and string:

```
<xs:element name="size">
  <xs:simpleType>
    <xs:union memberTypes="size_no size_string" />
  </xs:simpleType>
</xs:element>
```

The input rows contain items with a description and size. An item can have a numeric size, such as 42. Or, an item can have a size that is a string value, such as large, medium, or small.

The following table shows input rows with a numeric size and a string size:

Desc	Size
shoes	42
shirt	large

Create one port for the item size. Map the port as a string. The SOAP message contains the following elements:

```
<item>
   <desc>shoes</desc>
   <size>42</size>
</item>

<item>
   <desc>shirt</desc>
   <size>large</size>
</item>
```

CHAPTER 8

Web Service Administration

This chapter includes the following topics:

- Web Service Administration Overview, 81
- Web Service Properties Configuration , 81
- Web Service Security Management, 84
- Web Service Logs, 88
- Web Service Monitoring, 88

Web Service Administration Overview

After you deploy a web service to a Data Integration Service, you can configure web service security, configure the web service, view web service logs, and monitor web service requests. You must have the appropriate privileges in order to perform these tasks.

After you deploy a web service to a Data Integration Service, you can complete the following tasks:

- Configure web service security. Enable web service security and assign permissions on web services.
- Configure the web service. Configure the web service properties and the web service operation properties.
- View the web service logs. View Data Integration Service logs for a web service. View the web service run-time logs in the web service run-time logs directory.
- Monitor the web service. Use the Administrator tool or the Monitoring tool to monitor web service requests.

Web Service Properties Configuration

Configure the web service properties and web service operation properties for each web service that you deploy to Data Integration Service.

To view or edit the properties of a web service in the Administrator tool, select the **Application** view of the Data Integration Service, expand the application name in the top panel and select the web service. The properties appear in the **Properties** view.

To view or edit the web service operation properties in the Administrator tool, select the **Application** view of the Data Integration Service, expand the application name in the top panel, expand the web service and select the web service operation. The properties appear in the **Properties** view.

Web Service Properties

Web service properties include read-only general properties and properties to configure the settings that the Data Integration Service uses when it runs a web service.

When you expand a web service in the top panel of the Applications view, you can access web service operations contained in the web service.

The Applications view displays read-only general properties for web services and web service operations. Properties that appear in the view depend on the object type.

The following table describes the read-only general properties for web services and web service operations:

Property	Description
Name	Name of the selected object. Appears for all objects.
Description	Short description of the selected object. Appears for all objects.
Type	Type of the selected object. Appears for all object types.
Location	The location of the selected object. This includes the domain and Data Integration Service name. Appears for all objects.
WSDL URL	WSDL URL used to connect to the web service. Appears for web services.

The following table describes the configurable web service properties:

Property	Description
Startup Type	Determines whether the web service is enabled to run when the application starts or when you start the web service.
Trace Level	Level of error messages written to the run-time web service log. Choose one of the following message levels: - OFF. The DTM process does not write messages to the web service run-time logs. - SEVERE. SEVERE messages include errors that might cause the web service to stop running. - WARNING. WARNING messages include recoverable failures or warnings. The DTM process writes WARNING and SEVERE messages to the web service run-time log. - INFO. INFO messages include web service status messages. The DTM process writes INFO, WARNING and SEVERE messages to the web service run-time log. - FINE. FINE messages include data processing errors for the web service request. The DTM process writes FINE, INFO, WARNING and SEVERE messages to the web service run-time log. - FINEST. FINEST message are used for debugging. The DTM process writes FINEST, FINE, INFO, WARNING and SEVERE messages to the web service run-time log. - ALL. The DTM process writes FINEST, FINE, INFO, WARNING and SEVERE messages to the web service run-time log. Default is INFO.
Request Timeout	Maximum number of milliseconds that the Data Integration Service runs an operation mapping before the web service request times out. Default is 3,600,000.
Maximum Concurrent Requests	Maximum number of requests that a web service can process at one time. Default is 10.
Sort Order	Sort order that the Data Integration Service to sort and compare data when running in Unicode mode.

Property	Description
Enable Transport Layer Security	Indicates that the web service must use HTTPS. If the Data Integration Service is not configured to use HTTPS, the web service will not start.
Enable WS-Security	Enables the Data Integration Service to validate the user credentials and verify that the user has permission to run each web service operation.
Optimization Level	The optimizer level that the Data Integration Service applies to the object. Enter the numeric value that is associated with the optimizer level that you want to configure. You can enter one of the following numeric values: - 0. The Data Integration Service does not apply optimization. - 1. The Data Integration Service applies the early projection optimization method. - 2. The Data Integration Service applies the early projection, early selection, push-into, and predicate optimization methods. - 3. The Data Integration Service applies the cost-based, early projection, early selection, push-into, predicate, and semi-join optimization methods.
DTM Keep Alive Time	Number of milliseconds that the DTM instance stays open after it completes the last request. Web service requests that are issued against the same operation can reuse the open instance. Use the keep alive time to increase performance when the time required to process the request is small compared to the initialization time for the DTM instance. If the request fails, the DTM instance terminates. Must be an integer. A negative integer value means that the DTM Keep Alive Time for the Data Integration Service is used. 0 means that the Data Integration Service does not keep the DTM instance in memory. Default is -1.
SOAP Output Precision	Maximum number of characters that the Data Integration Service generates for the response message. The Data Integration Service truncates the response message when the response message exceeds the SOAP output precision. Default is 200,000.
SOAP Input Precision	Maximum number of characters that the Data Integration Service parses in the request message. The web service request fails when the request message exceeds the SOAP input precision. Default is 200,000.

Web Service Operation Properties

Configure the settings that the Data Integration Service uses when it runs a web service operation.

The following tables describes the configurable web service operation property:

Property	Description
Result Set Cache Expiration Period	The number of milliseconds that the result set cache is available for use. If set to -1, the cache never expires. If set to 0, result set caching is disabled. Changes to the expiration period do not apply to existing caches. If you want all caches to use the same expiration period, purge the result set cache after you change the expiration period. Default is 0.

Web Service Result Set Caching

Result set caching enables the Data Integration Service to use cached results for web service requests. Users that run identical queries in a short period of time may want to use result set caching to decrease the runtime of identical queries.

When you configure result set caching, the Data Integration Service caches the results of the DTM process associated with each web service request. The Data Integration Service caches the results for the expiration period that you configure. When an external client makes the same query or request before the cache expires, the Data Integration Service returns the cached results. If a cache does not exist or has expired, the Data Integration Service starts a DTM instance to process the request.

When the amount of data in the cache exceeds the maximum cache size memory, the Data Integration Service stores the result set in an encrypted cache file at `<Informatica_install_dir>/tomcat/bin/disTemp/<Service_Name>/<Node_Name>/`.

The Data Integration Service stores the result set cache for web services by user when the web service uses WS-Security. The Data Integration Service stores the cache by the user name that is provided in the username token of the web service request. When the Data Integration Service caches the results by user, the Data Integration Service only returns cached results to the user that sent the web service request.

Complete the following steps to configure result set caching in the Administrator tool:

1. Configure the result set cache properties in the Data Integration Service process properties.
2. Configure the cache expiration period in the web service operation properties. If you want the Data Integration Service to cache the results by user, enable WS-Security in the web service properties.

To disable result set caching for a web service request when the web service operation is configured to cache the result set, include the following syntax in the HTTP header of the SOAP request:
`WebServiceOptions.disableResultSetCache=true`

Web Service Security Management

An HTTP client filter, transport layer security, and message layer security can provide secure data transfer and authorized data access for a web service. When you configure message layer security, the Data Integration Service can pass credentials to connections.

You can configure the following security options for a web service:

HTTP Client Filter

If you want the Data Integration Service to accept requests based on the host name or IP address of the web service client, use the Administrator tool to configure an HTTP client filter. By default, a web service client running on any machine can send requests.

Message Layer Security

If you want the Data Integration Service to authenticate user credentials in SOAP requests, use the Administrator tool to enable WS-Security and configure web service permissions. The Data Integration Service can validate user credentials that are provided as a user name token in the SOAP request. If the user name token is not valid, the Data Integration Service rejects the request and sends a system-defined fault to the web service client. If a user does not have permission to execute the web service operation, the Data Integration Service rejects the request and sends a system-defined fault to the web service client.

Transport Layer Security (TLS)

If you want the web service and web service client to communicate using an HTTPS URL, use the Administrator tool to enable transport layer security (TLS) for a web service. The Data Integration Service that the web service runs on must also use the HTTPS protocol. An HTTPS URL uses SSL to provide a secure connection for data transfer between a web service and a web service client.

Pass-Through Security

If an operation mapping requires connection credentials, the Data Integration Service can pass credentials from the user name token in the SOAP request to the connection. To configure the Data Integration Service to pass credentials to a connection, use the Administrator tool to configure the Data Integration Service to use pass-through security for the connection and enable WS-Security for the web service.

Note: You cannot use pass-through security when the user name token includes a hashed or digested password.

Web Service Permissions

Permissions control the level of access that a user has to a web service when a web service requires user authentication.

Use the Administrator tool to configure permissions for a web service or a web service operation.

You can assign the following permissions to users and groups:

- Grant permission. Users can manage permissions on the web service objects using the Administrator tool or using the *infacmd* command line program.
- Execute permission. Users can send web service requests and receive web service responses.

The following table describes the permissions for each web service object:

Object	Grant Permission	Execute Permission
Web service	Grant and revoke permission on the web service and all web service operations within the web service.	Send web service requests and receive web service responses from all web service operations within the web service.
Web service operation	Grant, revoke, and deny permission on the web service operation.	Send web service requests and receive web service responses from the web service operation.

UserName Token in a SOAP Request

Web service clients must include a user name token header in the SOAP request when a web service requires user authentication. When a web service does not require user authentication, the Data Integration Service ignores the user name token header provided in the SOAP request.

The user name token element in a SOAP request can have one of the following password types:

- Plain text
- Hashed
- Digested

Note: You cannot use LDAP authentication when the user name token includes a hashed or digested password.

Include the user password in the Password element of the UsernameToken element. The Password element has a Type attribute to indicate the type of password security used.

Plain Text Password

Include a plain text password in the user name token header of a SOAP request when the user password does not need to be encrypted. The Data Integration Service can process plain text passwords in the UsernameToken element.

When the password is plain text, the UsernameToken element includes the following child elements:

Username element

Contains a user name in the 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 (/).

Password element

Contains the password in plain text. Set the Type attribute of the Password element to "PasswordText."

The following sample SOAP header shows an example of a UsernameToken element with a plain text password:

```
<soap:Header>
      <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">
         <wsse:UsernameToken wsu:Id="UsernameToken-14" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
            <wsse:Username>Administrator</wsse:Username>
            <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-1.0#PasswordText">Administrator</wsse:Password>
         </wsse:UsernameToken>
      </wsse:Security>
</soap:Header>
```

Hashed Password

Include a hashed password in the user name token header of a SOAP request when the user password must be encrypted. The Data Integration Service can process hashed passwords in the UsernameToken element.

When you use a hashed password, the UsernameToken element includes the following child elements:

Username element

Contains a user name in the 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. Set the Type attribute of the Password element to "PasswordText."

The following sample SOAP header shows an example of a UsernameToken element with a hashed password:

```
<soap:Header>
      <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">
         <wsse:UsernameToken wsu:Id="UsernameToken-14" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
            <wsse:Username>Administrator</wsse:Username>
            <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-1.0#PasswordText">Ntm58Cxf7SBOQAz3OlsTq1nv-D7</wsse:Password>
         </wsse:UsernameToken>
      </wsse:Security>
</soap:Header>
```

Digested Password

Include a digested password in the user name token header of a SOAP request when the user password is an encrypted password that is hashed with a nonce value and a time stamp. The Data Integration Service can process digested passwords in the UsernameToken element.

When you use a digested password, the UsernameToken element includes the following child elements:

Username element

Contains a user name that can be found in the 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 time stamp in the Created element. The password must be hashed with the SHA-1 hash function and encoded to Base64. For digested password security, set the Type attribute of the Password element to "PasswordDigest."

Nonce element

Contains a nonce value, which is a random value that can be used only once. By default, it is valid for 300 seconds 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 service 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.

Created element

Contains a time stamp value that indicates the time when the request was created. The time stamp uses the UTC format, `yyyy-MM-dd'T'HH:mm:ss.SSS'Z'`. For example: `2008-08-11T18:06:32.425Z`.

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, time stamp, and the digested password.

The following sample SOAP header shows an example of a UsernameToken element with a digested password:

```
<soap:Header>
      <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">
         <wsse:UsernameToken wsu:Id="UsernameToken-14" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
            <wsse:Username>Administrator</wsse:Username>
            <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-1.0#PasswordDigest">Ntm58Cxf7SBOQAz3OlsTq1nv-D7</wsse:Password>
            <wsse:Nonce EncodingType="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-soap-message-security-1.0#Base64Binary">zWELHdoAzNjQQ9xzlIwFZA==</
wsse:Nonce>
            <wsu:Created>2010-10-15T20:56:18.633Z</wsu:Created>
         </wsse:UsernameToken>
      </wsse:Security>
</soap:Header>
```

Web Service Logs

You can view web service logs in the Data Integration Service logs and in the web service run-time logs.

The Data Integration Service logs can contain web service messages such as, a web service status change or an error that prevents the web service from running. View the Data Integration Service logs on the **Logs** tab in the Administrator tool.

When the DTM process of the Data Integration Service runs the web service operation mapping, messages about the web service operation mapping appear in the web service run-time logs. The DTM process of the Data Integration Service creates a log file for each web service request. View the web service run-time logs in the `ws` directory within the Data Integration Service process log location. By default, view the web service run-time logs in the following directory: `<InformaticaInstallationDir>/tomcat/bin/disLogs/ws`

Web Service Trace Levels

In the Administrator tool, configure the **Trace Level** property to indicate the trace level of the messages that the DTM process writes to the web service run-time logs. You can configure the HTTP header in a web service request to override the trace level set for the web service.

Each web service request generates run-time logs based on the web service trace level configuration. To override the web service trace level for a web service request, include the following entry in the HTTP header of the web service SOAP request: `WebServiceOptions.traceLevel= <trace_level>`. For example, to use the HTTP header to set the trace level to FINE, use the following text: `WebServiceOptions.traceLevel= fine`

Web Service Monitoring

Monitor a web service to view its properties, run-time statistics, run-time reports, and information about each web service request.

You can monitor a web service in the following locations:

- Monitoring tool. In the Developer tool, click the **Menu** button in the **Progress** view and select **Monitor Jobs**. Select the Data Integration Service that runs the web service and click **OK**. The Monitoring tool opens.
- Administrator tool. To monitor web services in the Administrator tool, click the **Monitor** tab.

When you monitor a web service, you can view summary statistics or execution statistics for the service. The **Summary Statistics** view displays graphical information about web service distribution and state. The **Execution Statistics** view displays information about web services that are deployed in an application.

To monitor a web service, expand an application in the Navigator and select the **Web Services** folder. A list of web services appears in the contents panel. The contents panel shows properties about each web service, such as the name, description, and state.

When you select a web service, the details panel shows the following views:

- Properties view. The Properties view shows general properties and run-time statistics for a web service.
- Reports view. The Reports view shows monitoring reports about the selected web service.
- Operations view. The Operations view shows the name and description of each operation included in the web service. The view also displays properties, requests, and reports about each operation.

- Requests view. The Requests view shows properties about each web service request, such as request ID, user name, state, start time, elapsed time, and end time. You can filter the list of requests.

Properties View for a Web Service

The **Properties** view shows general properties and run-time statistics for a web service.

When you select a web service in the contents panel of the **Properties** view, you can view the general properties and monitoring statistics.

General Properties for a Web Service

You can view general properties about the web service, such as the name and type of object.

Statistics for a Web Service

You can view run-time statistics about web service requests during a specific time period. The **Statistics** section shows the number of completed, failed, and total web service requests.

Reports View for a Web Service

The **Reports** view shows monitoring reports about the selected web service.

When you monitor a web service, the **Reports** view shows reports about the web service. For example, you can view the Most Active WebService Client IP report to determine the IP addresses that received the most number of web service requests during a specific time period.

Operations View for a Web Service

The **Operations** view shows the name and description of each operation included in the web service. The view also displays properties, requests, and reports about each operation.

When you select a web service operation in the contents panel, the details panel shows the **Properties** view, **Requests** view, and **Reports** view.

Properties View for a Web Service Operation

The **Properties** view shows general properties and statistics about the selected web service operation. General properties include the operation name and type of object. The view also shows statistics about the web service operation during a particular time period. Statistics include the number of completed, failed, and total web service requests.

Requests View for a Web Service Operation

The **Requests** view shows properties about each web service operation, such as request ID, user name, state, start time, elapsed time, and end time. You can filter the list of requests. You can also view logs for the selected web service request.

Reports View for a Web Service Operation

The **Reports** view shows reports about web service operations.

Requests View for a Web Service

The **Requests** view shows properties about each web service request, such as request ID, user name, state, start time, elapsed time, and end time. You can filter the list of requests.

When you select a web service request in the contents panel, you can view logs about the request in the details panel. The details panel shows general properties and statistics about the selected web service request. Statistics include the number of completed, failed, and total web service requests.

You can also abort a web service request from the **Requests** view. To abort a web service request, select the workflow request and click **Actions** > **Abort Selected Request** in the contents panel.

CHAPTER 9

Web Service Consumer Transformation

This chapter includes the following topics:

- Web Service Consumer Transformation Overview, 91
- WSDL Selection, 94
- Web Service Consumer Transformation Ports, 94
- Web Service Consumer Transformation Input Mapping, 96
- Web Service Consumer Transformation Output Mapping, 99
- Web Service Consumer Transformation Advanced Properties, 102
- Filter Optimizations, 106
- Creating a Web Service Consumer Transformation, 108
- Web Service Consumer Transformation Example, 110

Web Service Consumer Transformation Overview

The Web Service Consumer transformation connects to a web service as a web service client to access or transform data. The Web Service Consumer transformation is a multiple-group transformation.

A web service uses open standards, such as SOAP, WSDL, and XML. SOAP is the communications protocol for web services. The web service client request and the web service response are SOAP messages. A WSDL is an XML schema that describes the protocols, formats, and signatures of the web service operations.

Web service operations include requests for information, requests to update data, or requests to perform tasks. For example, the Web Service Consumer transformation sends a SOAP request to run a web service operation called getCustomerOrders. The transformation passes a customer ID in the request. The web service retrieves the customer and order information. The web service returns the information to the transformation in a SOAP response.

The Web Service Consumer transformation connects to a web service using an endpoint URL defined in the WSDL, in a web services connection, or in an endpoint URL input port. You enable security for web services in a web services connection.

SOAP Messages

The Web Service Consumer transformation uses Simple Object Access Protocol (SOAP) to exchange information with the web services provider and to request web services. SOAP defines the format for web service request and response messages.

When you transform data with a Web Service Consumer transformation, the transformation generates a SOAP request and connects to the web service. The transformation connects to the web service using an endpoint URL defined in the WSDL object, in a web services connection, or in an endpoint URL input port. The SOAP request contains the information that the web service requires to run the requested operation. The web service operation returns data to the transformation in a SOAP response. The transformation maps data from the SOAP response and returns the data in output ports.

The Web Service Consumer transformation encodes the SOAP message headers in ISO-8859-1.

The transformation can process SOAP messages with document/literal encoding. The document/literal style requires an XML schema to describe the SOAP message. SOAP messages are formed from the XML. When a SOAP message contains multiple-occurring elements, the groups of elements form levels in the XML hierarchy. The groups are related when one level is nested within another.

A SOAP request message can contain hierarchical data. For example, the Web Service Consumer transformation sends a request to add customer orders to a sales database. The transformation passes two groups of data in a SOAP request message. One group contains a customer ID and name, and the other group contains order information. The order information occurs multiple times.

A SOAP response message can contain hierarchical data. For example, the Web Service Consumer transformation generates a SOAP request for customer orders. The web service returns an order header and multiple-occurring order detail elements in the SOAP response.

WSDL Files

A WSDL file contains a description of the data to be passed to the web service so that the sender and the receiver understand the data to exchange. You must import a WSDL file to the repository before you can create a Web Service Consumer transformation.

The WSDL describes the operations to perform on the data and a binding to a protocol or transport, so that the web service consumer can send the request message in the correct format. The WSDL describes the network address to connect to the web service.

The WSDL includes information about how to encode SOAP request and response messages. SOAP encoding determines the format of the SOAP message body. It describes the format for request and response messages that the web service uses to communicate to the web service consumer. Web service developers can use a variety of toolkits to create web services. Toolkits support different ways of encoding SOAP messages.

The Web Service Consumer transformation supports the document/literal SOAP encoding style. You can use WSDL 1.1 with the Web Service Consumer transformation. You cannot use WSDL attachments such as MIME, DIME, and MTOM messages.

Operations

A web service contains an operation for each action supported by the web service.

For example, a web service can have an operation named getcustomerid that receives a customer name and responds with the customer details. The operation input includes an element for the customer name. The operation output includes elements for customer details based on the customer name.

When you configure a Web Service Consumer transformation, you define how the transformation maps data to the operation input and how the transformation maps data from the operation output. You configure the following information in the transformation:

Input mapping

Define how to map the transformation input ports to the web service operation input nodes. The operation input defines the elements in the SOAP request for the operation.

Output mapping

Define how to map the web service operation output nodes to the transformation output ports. The operation output defines the elements in a SOAP response for the operation.

Web Service Security

You enable security for web services in a web services connection. You can configure the following types of security:

Web Service Security

The Data Integration Service can include a web service security header when it sends a SOAP request to the web service provider. The web service security header contains authentication information so the web service provider can authenticate the Data Integration Service.

The Web Service Consumer transformation supplies the user name token. The Data Integration Service creates a separate security SOAP header in the SOAP request and passes the request to the web service provider.

You can use the following types of web service security in a web services connection:

- PasswordText. The Data Integration Service does not change the password in the WS-Security SOAP header.
- PasswordDigest. The Data Integration Service combines the password with a nonce and a time stamp. The Data Integration Service applies a SHA hash on the password, encodes it in base64 encoding, and uses the encoded password in the SOAP header.

Transport layer security

Security implemented on top of the transport layer (TCP layer) of TCP/IP using Secure Sockets Layer (SSL). Web services use Hypertext Transfer Protocol over SSL (HTTPS) as a web address for secure message transport. Web Service Consumer transformations can use TLS 1.2, TLS 1.1, or TLS 1.0. You can use the following authentication with transport layer security: HTTP authentication, proxy server authentication, and SSL certificates.

SSL authentication

You can use SSL authentication when you connect through the HTTPS protocol.

You can use the following types of SSL authentication:

- One-way SSL authentication
- Two-way SSL authentication

HTTP authentication

You can use HTTP authentication when you connect through the HTTP protocol.

You can use the following HTTP authentication methods:

- Basic authentication
- Digest authentication

- NT LAN Manager (NTLM) authentication

WSDL Selection

Before you create a Web Service Consumer transformation, you must import a WSDL file into the model repository. The WSDL defines the operation signature of the web service you want to run. When you import a WSDL, the Developer tool creates a physical data object that you can reuse for other transformations.

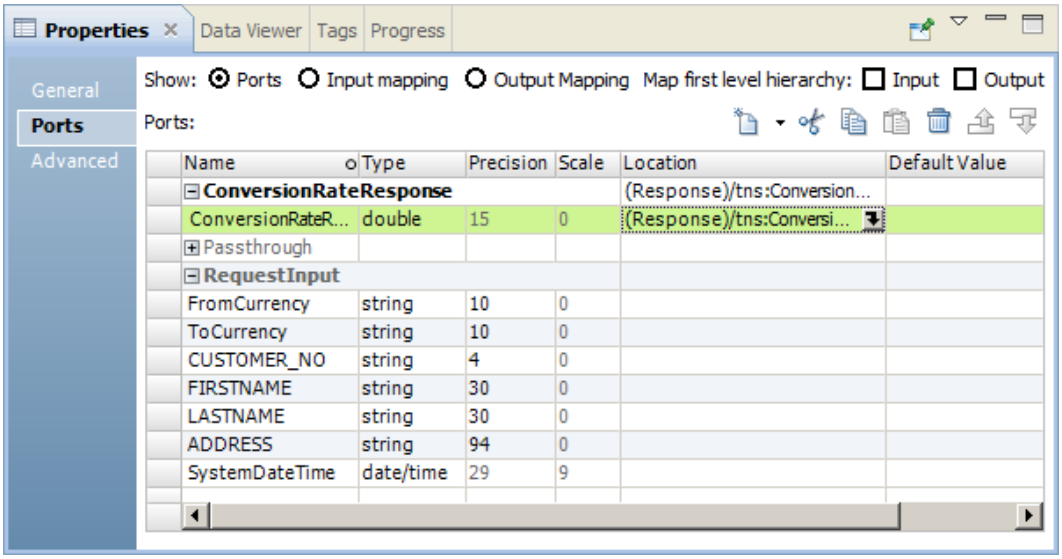
A WSDL can define multiple operations. When you create a Web Service Consumer transformation, select which operation you want to run. You can view the operation input and the operation output hierarchies in the Web Service Consumer transformation. The hierarchies define the structure of the SOAP request message and the SOAP response message.

You can also import a WSDL with one-way input operation. You must create dummy output ports when you import a WSDL with one-way input operation.

Web Service Consumer Transformation Ports

When you view the transformation ports, show the ports if you do not need to view the operation hierarchy. When you show the ports, you can define groups, define ports, and map nodes from the operation output to the output ports.

The following figure shows the ports for a non-reusable Web Service Consumer transformation:



A Web Service Consumer transformation can have multiple input groups and multiple output groups. When you create ports, create groups and add the ports to the groups. Define the ports in a group hierarchy based on the structure of the operation input or operation output hierarchy. Add a key to relate a child group to a parent group. All groups except the lowest group in the hierarchy must have primary keys. All groups in the hierarchy except the root group must have foreign keys.

The transformation has a root input group named RequestInput. You must add a primary key to the root input group. The key must be string, bigint, or integer.

You can add additional pass-through ports to the root input group. Pass-through ports pass data through the transformation without modifying the data. The pass-through port can occur one time in the input data. You can add the pass-through port to any output group. Associate the output port to the input port. The input value that you pass through a SOAP request repeats in the output rows from the SOAP response.

You can also add HTTP headers, cookie ports, a dynamic URL port, and ports for web service security authentication to the root input group. Data in the root group occurs one time.

To map an operation output node to an output port, click the field in the **Location** column and expand the hierarchy in the **Select Location** dialog box. Then, choose a node from the hierarchy.

HTTP Header Input Ports

A web service might require additional HTTP headers. You can create input ports in the root input group to pass the additional header information to the web service provider.

To add an HTTP header and an HTTP port, select the root input group and click the arrow next to the **New** button. Then, click **HTTP Header**. Enter a header name and port name.

You can create multiple HTTP headers.

Other Input Ports

You can add predefined input ports to the Web Service Consumer transformation.

You can add the following predefined input ports:

Cookie port

You can configure the Web Service Consumer transformation to use cookie authentication. The remote web service server tracks the web service consumer users based on the cookies. You can increase performance when a mapping calls a web service multiple times.

When you project the cookie port to a web service request message, the web service provider returns a cookie value in the response message. You can pass the cookie value to another transformation downstream in the mapping or you can save the cookie value in a file. When you save the cookie value in a file, you can configure the cookie as input to the Web Service Consumer transformation.

You can project the cookie output port to any of the Web Service Consumer transformation output groups.

Endpoint URL port

The Web Service Consumer transformation connects to a web service using an endpoint URL. You can define the endpoint URL in the WSDL file, in a web services connection, or in an endpoint URL input port. When the transformation receives the URL dynamically in a port, the Data Integration Service overrides the URL defined in the WSDL file or in the web services connection.

The Web Service Consumer transformation can have one URL port value for each web service request. Add an Endpoint URL port to the root input group.

WS-Security ports

You enable web service security in a web services connection. When you enable web service security, you must define the user name and password in a web services connection or in WS-Security input ports.

When you add WS-Security ports, you pass the user name and the password through input ports in the transformation. When the transformation receives the user name and password dynamically in ports, the Data Integration Service overrides the values defined in the web services connection.

Note: A web services connection has one user name and password for HTTP and WS-Security authentication.

To add predefined input ports, click the root input group in the **Ports** area. Click the arrow next to the **New** button and then click **Other Ports**. Choose the ports to add.

Web Service Consumer Transformation Input Mapping

When you view the transformation ports, show the input mapping to view the operation input hierarchy. When you show the input mapping, you can define input groups, define input ports, and map input ports to operation input nodes.

The input mapping includes the following areas:

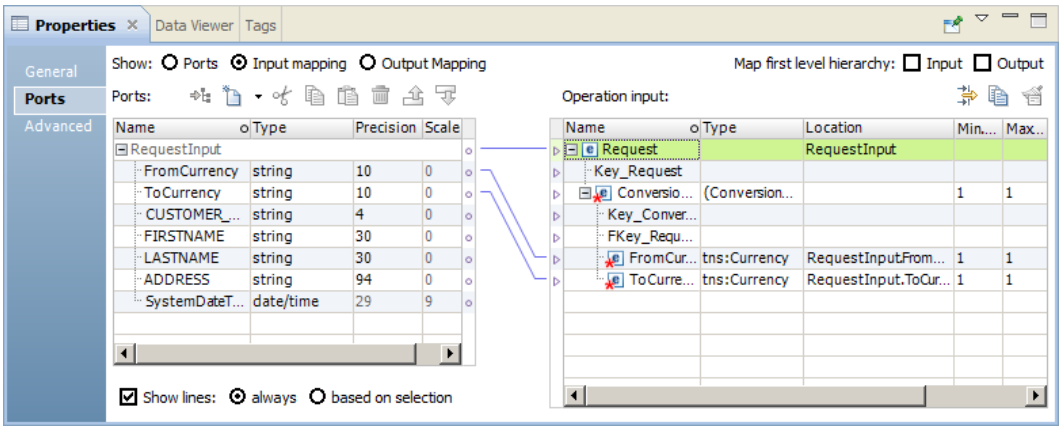
Ports

Create the transformation input groups and input ports in the **Ports** area.

Operation Input

The **Operation Input** area shows the nodes in the SOAP request message that the Web Service Consumer transformation sends to the web service. The WSDL data object that you use to create the transformation defines the operation input hierarchy.

The following figure shows the input mapping for a non-reusable Web Service Consumer transformation:



After you create input ports, map the input ports from the **Ports** area to the nodes in the **Operation Input** area. When you map an input port to a node in the operation input, the location of the port appears in the **Location** column in the **Operation Input** area.

The Developer tool maps nodes in the first level of the operation input to input ports when you choose to map the first level of the input hierarchy. The Developer tool also creates the ports to perform the mapping. If the first level of the hierarchy contains a multi-occurring parent node with one or more multi-occurring child nodes, the Developer tool does not map the first level of the hierarchy.

You can map XML data from one string or text input port to the entire SOAP request message. When you map XML data to the entire SOAP request, you cannot map ports to nodes in the operation input.

You can choose to view the lines that connect the input ports to the nodes in the operation input.

Rules and Guidelines to Map Input Ports to Nodes

Review the following rules when you map input ports to nodes in the operation input hierarchy:

- You can map an input port to one node in the hierarchy. You can map the same port to any number of keys in the hierarchy.
- The input port and the node must have compatible data types.
- You can map ports from one input group to multiple hierarchy levels in the operation input.
- You must map input ports to the keys in the operation input. Any port that you map to a key must be a string, integer, or bigint datatype. Map data to the keys in all levels in the operation input above the hierarchy level that you are including in the SOAP message. Include the foreign keys for all levels above and including the level you are mapping.

  Note: You do not have to map input ports to keys if you are mapping only the lowest level of the operation input hierarchy.
- You can map multiple string, bigint, or integer input ports to a key in the **Operation Input** area to create a composite key. When you click the **Location** field for a composite key, you can reorder the input ports or remove one of the ports.

Customize View Options

You can change the operation input hierarchy to show keys in the **Operation Input** area. You can also show grouping constructs that define how to order nodes.

Click the **Customize View** button ( ) in the **Operation Input** area. Enable any of the following options:

Sequence, Choice, and All

Show a line that indicates whether an element definition is sequence, choice, or all.

Nodes in an all group must all be included in the SOAP message.

Nodes in a sequence group must be in the order specified in the WSDL.

At least one node in a choice group must appear in the SOAP message.

Keys

View the keys in the **Operation Input** area. The **Operation Input** area includes keys for each group. You can add a key to an input port in the **Ports** area.

Mapping Input Ports to the Operation Input

When you show the transformation input mapping, you can define input groups, define input ports, and map input ports to operation input nodes.

1. Open a Web Service Consumer transformation.
2. To show the transformation input mapping, use one of the following methods:
   - For a reusable transformation, click the **Overview** view. Choose to show the input mapping.
   - For a non-reusable transformation, click the **Ports** tab in the **Properties** view. Choose to show the input mapping.

3. Define a primary key for the root input group.
4. To add an input group or port to the **Ports** area, use one of the following methods:

Option	Description
Drag a node	Drag a group node or a child node in the **Operation Input** area to an empty column in the **Ports** area. If the node is a group node, the Developer tool adds a group without ports.
Manually add a group or port	To add a group, click the arrow next to the **New** button and then click **Group**. To add a port, click the arrow next to the **New** button and then click **Field**.
Drag a port from another transformation	In the editor, drag a port from another transformation to the Web Service Consumer transformation.
Copy a port	Select ports from another transformation and copy them to the **Ports** area. To copy ports, you can use keyboard shortcuts or you can use the copy and paste buttons in the Developer tool.
Select **Map first level of hierarchy**	Select **Map first level of hierarchy**. The Developer tool maps nodes in the first level of the operation input to input ports and groups. The Developer tool also creates the input ports and groups to perform the mapping.

5. If you manually create a port or you copy a port from another transformation, click the **Location** column in the **Operation Input** area and choose a port from the list.
6. To map input ports as a composite key, use one of the following methods:

Option	Description
Drag input ports	Select two or more input ports and drag them to a key in the operation input hierarchy.
Select input ports from the **Select Location** dialog box	Click the **Location** column of a key in the operation input hierarchy and then select the input ports.

7. To clear the locations of nodes, use one of the following methods:

Option	Description
Click **Clear**	Select one or more nodes in the **Operation Input** area and click **Clear**.
Delete the lines that connect ports to nodes	Select one or more lines that connect the input ports to the nodes in the operation input and press **Delete**.

8. If the associated WSDL data object includes anyType elements, any elements, anyAttribute attributes, derived type elements, or substitution groups, choose objects in the **Operation Input** area. In the **Type** column for a node, click **Choose** and then choose one or more types, elements, or attributes from the list.
9. To map XML data from a string or text input port to the complete SOAP request, right-click the port and select **Map as XML**.

Web Service Consumer Transformation Output Mapping

When you view the transformation ports, show the output mapping to view the operation output hierarchy. When you show the output mapping, you can define output groups, define output ports, and map operation output nodes to output ports.

The output mapping includes the following areas:

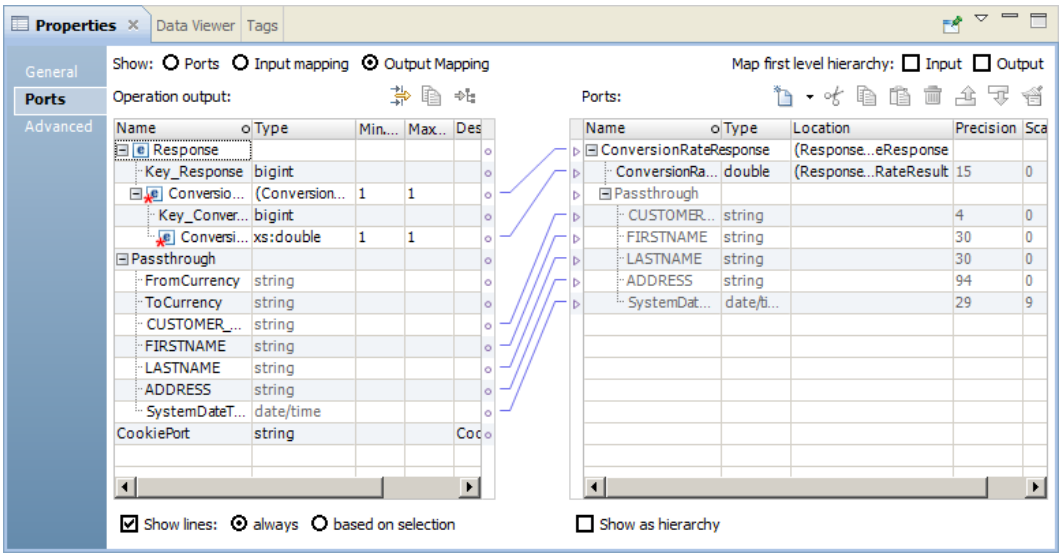
Operation Output

The **Operation Output** area shows the nodes in the SOAP response message that the web service returns to the Web Service Consumer transformation. The WSDL data object that you use to create the transformation defines the operation output hierarchy.

Ports

Create the transformation output groups and ports in the **Ports** area.

The following figure shows the output mapping for a non-reusable Web Service Consumer transformation:



After you create output ports, map the nodes from the **Operation Output** area to the ports in the **Ports** area. When you map a node from the operation output to an output port, the location of the node appears in the **Location** column in the **Ports** area.

The Developer tool maps nodes in the first level of the operation output to output ports when you choose to map the first level of the output hierarchy. The Developer tool also creates the ports to perform the mapping. If the first level of the hierarchy contains a multi-occurring parent node with one or more multi-occurring child nodes, the Developer tool does not map the first level of the hierarchy.

You can choose to display the output ports in a hierarchy. Each child group appears under the parent group. You can also choose to view the lines that connect the nodes in the operation output to the output ports.

If the associated WSDL data object is deleted from the repository, the Developer tool retains the location of the operation nodes in the output mapping. When you show the output mapping, the **Ports** area still displays the location of the operation nodes in the **Location** column for the output ports. If you associate another WSDL with the transformation, the Developer tool checks whether each location is valid. The Developer tool clears the location of operation nodes in the **Ports** area of the output mapping if the location is no longer valid.

Rules and Guidelines to Map Nodes to Output Ports

Review the following rules when you map nodes in the operation output hierarchy to output ports:

- The operation output node and the output port must have compatible datatypes.
- You cannot map a node to more than one output port in a group.
- Each output port must have a valid location, unless the port is a pass-through port.
- If you drag a multiple-occurring child node to an empty output port, you must relate the group to other output groups. When you select a group, the Developer tool creates keys to relate the groups.
- When you drag a multiple-occurring element into a group that contains the parent element, you can configure the number of child element occurrences to include. Or, you can replace the parent group with the multiple-occurring child group in the transformation output.

Mapping the SOAP Message as XML

You can map the complete SOAP message as XML instead of returning the data to separate output ports.

When you map the SOAP message as XML, the Data Integration Service returns the complete SOAP message in one port. Do not create output ports.

To map the complete message, right-click the root group in the **Operation Output** area. Select **Map as XML**.

The Developer tool creates a string output port. The precision is 65535 bytes.

Customize View Options

You can change the operation output hierarchy to show the cookie ports, pass-through ports, and keys in the **Operation Output** area. You can also show grouping constructs that define how to order nodes.

Click the **Customize View** button in the **Operation Output** area. Enable any of the following options:

Sequence, Choice, and All

Show a line that indicates whether an element definition is sequence, choice, or all.

Nodes in an all group must all be included in the SOAP message.

Nodes in a sequence group must be in the order specified in the WSDL.

At least one node in a choice group must appear in the SOAP message.

Keys

View the keys in the **Operation Output** area. The **Operation Output** area includes keys for each group. You can add a key to an output port in the **Ports** area.

Pass-through Ports

The **Operation Output** area shows the pass-through ports. Pass-through ports are ports that pass data through the transformation without changing the data. You can project pass-through ports from the operation output to any of the Web Service Consumer transformation output groups. A pass-through port receives data one time, so the port is at the root level in the SOAP messages.

Cookie Ports

Shows the cookie port. When you configure cookie authentication, the remote web service server tracks the web service consumer users based on the cookies. When you project a web service cookie in the request message, the web service returns a cookie in the response message. You can project the cookie from the operation output to any of the Web Service Consumer transformation output groups.

Mapping the Operation Output to Output Ports

When you show the transformation output mapping, you can define output groups, define output ports, and map operation output nodes to output ports.

1. Open a Web Service Consumer transformation.
2. To show the transformation output mapping, use one of the following methods:
   - For a reusable transformation, click the **Overview** view. Choose to show the output mapping.
   - For a non-reusable transformation, click the **Ports** tab in the **Properties** view. Choose to show the output mapping.
3. To add an output group or port to the **Ports** area, use one of the following methods:

Option	Description
Drag a node	Drag a group node or a child node in the **Operation Output** area to an empty column in the **Ports** area. If the node is a group node, the Developer tool adds a group without ports.
Manually add a group or port	To add a group, click the arrow next to the **New** button and then click **Group**. To add a port, click the arrow next to the **New** button and then click **Field**.
Drag a port from another transformation	In the editor, drag a port from another transformation to the Web Service Consumer transformation.
Copy a port	Select ports from another transformation and copy them to the **Ports** area. To copy ports, you can use keyboard shortcuts or you can use the copy and paste buttons in the Developer tool.
Select **Map first level of hierarchy**	Select **Map first level of hierarchy**. The Developer tool maps nodes in the first level of the operation output to output ports and groups. The Developer tool also creates the output ports and groups to perform the mapping.

4. If you manually create a port or you copy a port from another transformation, click the **Location** column in the **Ports** area and choose a node from the list.
5. To clear the locations of ports, use one of the following methods:

Option	Description
Click **Clear**	Select one or more ports in the **Ports** area and click **Clear**.
Delete the lines that connect nodes to ports	Select one or more lines that connect the nodes in the operation output to the output ports and press **Delete**.

6. If the associated WSDL data object includes anyType elements, any elements, anyAttribute attributes, derived type elements, or substitution groups, choose objects in the **Operation Output** area. In the **Type** column for a node, click **Choose** and then choose one or more types, elements, or attributes from the list.
7. To map the complete SOAP response message as XML, right-click the root group in the **Operation Output** area and select **Map as XML**.

Web Service Consumer Transformation Advanced Properties

The Web Service Consumer transformation advanced properties include the tracing level, generic fault ports, web services connection, and concurrent web service request messages.

You can define the following advanced properties for the Web Service Consumer transformation on the Advanced tab:

Tracing Level

Amount of detail that appears in the log for this transformation. You can choose terse, normal, verbose initialization, or verbose data. Default is normal.

SOAP Action

Overrides the SOAP action value defined in the WSDL with a constant value for the Web Service Consumer transformation.

Enable Generic SOAP Fault Handling

Returns fault messages that are not defined in the WSDL. Creates output ports in a GenericFault output group to handle fault codes and messages.

The following table describes the fault output ports for SOAP 1.1 and SOAP 1.2:

Fault Output Port for SOAP 1.1	Fault Output Port for SOAP 1.2	Description
Fault Code	Code*	Returns a fault identification code.
Fault String	Reason*	Returns an explanation of the error in a fault message.
Fault Detail	Detail	Returns custom information that the web service provider passes to the Web Service Consumer transformation in a generic fault message.
Fault Actor	Role	Returns information about the object that caused the fault to happen.
-	Node	Returns the URI of the SOAP node that generated the fault.
* The Code and Reason output ports are hierarchical.		

Note: You can expand the Code fault output port to extract the SubCode fault output port upto one level.

Enable HTTP Error Handling

Returns any HTTP error from the web service. Creates an HTTP error output port in the GenericFault output group.

Treat Fault as Error

Adds fault messages to the mapping log. When a fault occurs, the Data Integration Service increments the error count for the mapping. Disable this property to allow early selection and push-into optimization. Default is enabled.

Connection

Identifies the web services connection object to connect to the web service. Create the web services connection in the Developer tool. Edit the web services connection in the Developer tool or the Administrator tool. When you configure a web services connection, configure the endpoint URL, the type of security that the web service requires, and a connection timeout period.

The Web Service Consumer transformation connects to a web service using an endpoint URL. You can define the endpoint URL in the WSDL file, in a web services connection, or in an endpoint URL input port.

Use the following guidelines to determine when to configure a web services connection:

- Configure a connection if you want to use an endpoint URL that differs from the URL in the WSDL file and if you are not using an endpoint URL input port.
- Configure a connection if the web service you connect to requires web service security, HTTP authentication, or SSL certificates.
- Configure a connection if you want to change the default connection timeout period.

Note: You can associate a WSDL data object in the repository with a web services connection. The associated connection becomes the default connection for each Web Service Consumer transformation that you create from that WSDL.

Enable Compression

Enables coding of SOAP requests with the GZIP compression method and enables decoding of SOAP responses with GZIP or deflate.

XML Schema Validation

Validates the SOAP response message at run time. Select **Error on Invalid XML** or **No Validation**.

Sorted Input

Enables the Data Integration Service to generate output without processing all of the input data. Enable sorted input when the input data is sorted by the keys in the operation input hierarchy.

Push-into Optimization

Enables push-into optimization. Click the **Open** button in the **Push-Into Optimization** property to select filter ports that receive filter values. For each filter port, choose the output port that contains the filtered column in the web service response.

Has Side Effects

Checkbox that indicates that the web service performs any function besides returning rows. The Web Service Consumer transformation has a side effect if the web service, in addition to returning a rows, modifies an object or interacts with other objects or functions. The web service might modify a database, add to a total, raise an exception, write an email, or call other web services with side effects. Disable the **Has Side Effects** property in order to allow push-into optimization or early selection optimization. Default is enabled.

Enable Concurrency

Enables the Web Service Consumer transformation to create multiple concurrent connections to a web service so that it can send multiple web service requests in parallel. When you enable the Web Service Consumer transformation to create multiple concurrent connections to the web service, you can set the total memory consumption limit and the number of concurrent connection limits.

The following table describes the options:

Options	Description
Enable concurrency	Creates multiple concurrent connections to a web service.
Concurrency Connection Limit	The number of concurrent web service connections. Default is 20.
Total Concurrency Memory Limit (in MB)	The total memory allocation limit for all the concurrent connections. Default is 100 MB.

Web Service Error Handling

You can configure the Web Service Consumer transformation to pass SOAP faults and HTTP errors downstream in a mapping. You can increment the error count when a fault occurs. Configure web service error handling in the transformation advanced properties.

A web service returns either a response message or it returns a fault. A fault is an error. The web service can generate different faults based on the errors that occur.

The Web Service Consumer transformation can return the following types of faults:

SOAP faults

SOAP errors that the WSDL defines. Configure output error ports that return the faults in the web service response message. For a SOAP 1.1 binding, the Data Integration Service returns the fault message, fault code, fault string, and fault actor elements for the fault. For a SOAP 1.2 binding, the Data Integration Service returns the fault message, code, reason, node, and role elements for the fault.

Generic SOAP faults

The web service generates generic SOAP faults at run time. The fault elements are different for a SOAP 1.1 binding and a SOAP 1.2 binding. The WSDL does not define generic SOAP faults. Generic SOAP faults include authentication failures and SOAP request errors.

HTTP errors

The Developer tool adds the HTTP fault output port when you enable HTTP error handling in the transformation. The Data Integration Service returns HTTP errors from the web service in a single string port. An HTTP error includes an error code and a message.

If the SOAP response from the web service has XML data that is not valid, the Web Service Consumer transformation returns an error.

You can configure whether to treat SOAP faults as errors. When you enable Treat Fault as Error and a SOAP fault occurs, the Data Integration Service increments the error count for the mapping. The fault appears in the message log.

Message Compression

When you enable SOAP message compression, the Web Service Consumer transformation compresses web service request messages and receives compressed web service response messages.

The Web Service Consumer transformation encodes the SOAP request with GZip compression. The transformation accepts a response message encoded with the GZip or the deflate compression.

When the Data Integration Service receives the response from the web service, it checks the Content-Encoding HTTP header in the SOAP message and it decodes the message.

The default is no compression encoding. The web service does not compress the SOAP response.

The following table shows the headers in the request and response messages when compression is on or off:

Compression	Header
On	Content-Encoding header: GZip Accept-Encoding header: GZip, deflate
Off	Empty Content-Encoding header Empty Accept-Encoding header

Sometimes a web service encodes a response message with a default compression. The Web Service Consumer transformation decodes the message if it is GZip or deflate encoded. The Web Service Consumer transformation logs a message to the mapping log if the web service encodes the response message unexpectedly.

Enable compression in the transformation advanced properties.

Concurrency

You can enable the Web Service Consumer transformation to create multiple concurrent connections to a web service so that it can send multiple web service requests in parallel.

For example, while querying bank information, you can configure the Web Service Consumer transformation for concurrency so that multiple rows are sent in parallel. If there are 20 input rows, you can send 20 requests concurrently for faster processing.

When you enable concurrency in the Web Service Consumer transformation, you can configure the total memory consumption limit.

When you enable concurrency in the Web Service Consumer transformation, you can configure the number of concurrent web service connections.

Rules and Guidelines for Concurrency

Use the following rules and guidelines while using concurrency:

- Concurrency supports sorted input rows as multiple concurrent connections to a web service. Ordered output rows are not supported.
- Use concurrency if the data set is more than 100 rows.
- It is advisable not to increase the number of concurrent web service connections. The number of concurrent web service connections is linked to the number of sockets used by the operating system. Increasing the number of sockets is expensive.
- Use systems that have multi-core processors with a minimum of 100 MB of RAM for optimal performance while using the concurrency feature.
- Concurrency memory limit represents the memory consumed by concurrent work flows while invoking web services.
- When you enable concurrency in the Web Service Consumer transformation, you can configure the memory consumption limit. Ensure that the memory consumption is not more than the physical RAM on the server.

Best Practices for Concurrency

For optimal performance while using concurrency, adhere to the following best practices:

- Avoid changing the default values of the total concurrency memory limit and concurrency connection limit.
- Avoid using concurrency for data sets of less than 100 rows.
- Avoid pass-through ports in the mapping while using concurrency.

Filter Optimizations

Filter optimization increases performance by reducing the number of rows that pass through the mapping. The Data Integration Service can apply the early selection optimization or push-into optimization.

When the Data Integration Service applies a filter optimization method, it moves a filter as close to the source as possible in a mapping. If the Data Integration Service cannot move a filter before a transformation in a mapping, it might be able to push the filter logic into a transformation.

Enabling Early Selection Optimization with the Web Service Consumer Transformation

Enable early selection optimization for the Web Service Consumer transformation if the transformation does not have side effects and it does not treat faults as errors.

1. Open the Web Service Consumer transformation **Advanced Properties** view.
2. Clear **Treat Fault as Error**.
3. Clear **Has Side Effects**.

Push-Into Optimization with the Web Service Consumer Transformation

You can configure push-into optimization with the Web Service Consumer transformation when the transformation is in a virtual table in an SQL data service.

The mapping calls the web service to retrieve a set of data or a subset of the data based on the statements in the end-user SQL query. The end-user SQL query contains an optional filter condition.

With push-into optimization, the Web Service Consumer transformation receives the filter value in a filter port. The filter port is an unconnected input port that you identify as a filter port when you configure push-into optimization. The filter port has a default value that ensures that the web service returns all rows if the end-user query contains no filter. The filter port is not a pass-through port.

Note: The filter field must be part of the root group in the web service request.

When you configure a filter port, you identify an output port in the Web Service Consumer transformation that receives the column data from the web service response. For example, if the filter port is an input port named EmployeeID, the output port from the response might be a port named EmployeeNum. The Developer tool needs to associate the input filter port and an output port in order to push the filter logic from the virtual table read to the web service consumer request. The input ports for a web service request are usually different than the output ports from the web service response.

The filter field cannot be a pass-through port. When you configure a filter port, the default value of the port changes to the value of the filter condition, so the pass-though output port value changes. A filter based on the output pass-through port returns unexpected results.

You can push multiple filter expressions to the Web Service Consumer transformation. Each filter condition must be the following format:

```
<Field> = <Constant>
```

The filter conditions must be joined by AND. You cannot join the conditions with an OR.

Push-Into Optimization with Web Service Consumer Transformation Example

An SQL data service returns orders for all customers or it returns orders for a specific customer based on the SQL query it receives from the user.

The data service contains a logical data object with the following components:

Customer table

An Oracle database table that contains customer information.

Web Service Consumer transformation

A transformation that calls a web service to retrieve the latest orders for customers. The Web Service Consumer transformation has input ports for customerID and orderNum. The transformation has pass-through ports that contain customer data that it receives from the Customer table. The orderNum port is the filter port and is not connected. orderNum has the default value "*". When the web service receives this value in the web service request, it returns all orders.

Orders virtual table

A virtual table that receives the customer and order data from the web service. The end-user queries this table. Orders contains a customer column, orderID column, and customer and order data.

The end-user passes the following SQL query to the SQL data service:

```
SELECT * from OrdersID where customer = 23 and orderID = 56
```

The Data Integration Service splits the query to optimize the mapping. The Data Integration Service uses early selection optimization and moves the filter logic, `customer = 23`, to the Customer table read. The Data Integration Service uses push-into optimization and pushes the filter logic, `orderID = 56`, into the Web Service Consumer transformation filter port. The Web Service Consumer transformation retrieves ordersID 56 for customer 23.

Enabling Push-Into Optimization with the Web Service Consumer Transformation

Enable push-into optimization for the Web Service Consumer transformation if the transformation does not have side effects and it does not treat faults as errors.

1. Open the Web Service Consumer transformation **Advanced Properties** view.
2. Clear **Treat Fault as Error**.
3. Clear **Has Side Effects**.
4. Click the **Open** button in the **Push-Into Optimization** property.
5. Choose the filter port name in the Optimized Input dialog box.
   You can choose multiple filter ports.
6. Click the **Output** column.

7. For each filter port, choose the output port that contains the filtered column in the web service response.
8. Enter a default value for each filter port.

   Note: You cannot configure a default value for a Web Service Consumer port unless it is a filter port.

Creating a Web Service Consumer Transformation

You can create a reusable or non-reusable Web Service Consumer transformation. Reusable transformations can exist in multiple mappings. Non-reusable transformations exist within a single mapping.

You can create Web Service Consumer transformations for a SOAP 1.1 binding and a SOAP 1.2 binding from a single WSDL object.

1. To create a transformation, use one of the following methods:

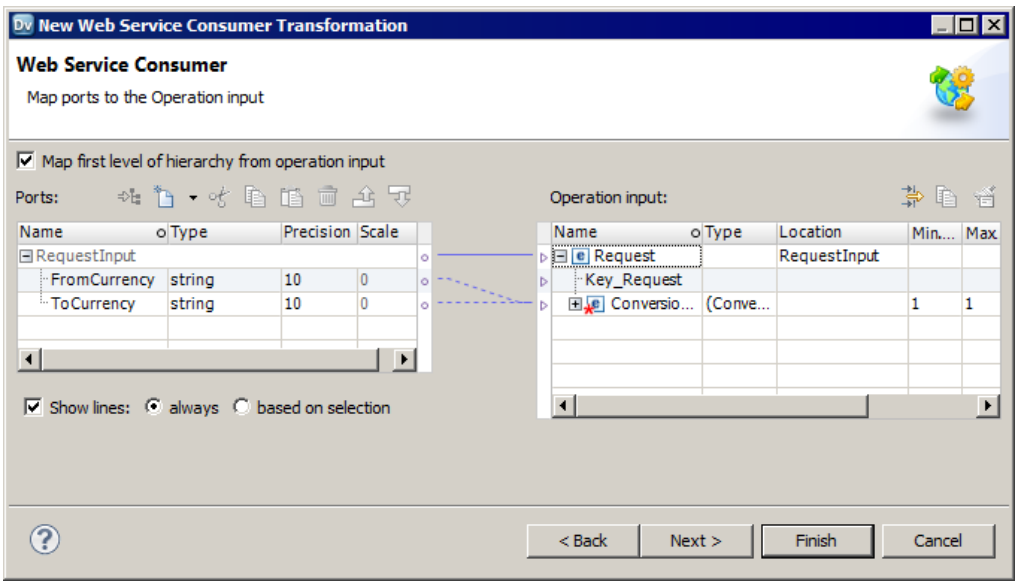
Option	Description
Reusable	Select a project or folder in the **Object Explorer** view. Click **File** > **New** > **Transformation**. Select the Web Service Consumer transformation and click **Next**.
Non-reusable	In a mapping or mapplet, drag a Web Service Consumer transformation from the Transformation pallette to the editor.

   The **New Web Service Consumer Transformation** dialog box appears.
2. Browse and select a WSDL data object to define the web service request and response messages.

   If the WSDL is not in the repository, you can import the WSDL from the New Web Service Consumer Transformation dialog box.
3. Browse and select an operation from the WSDL.

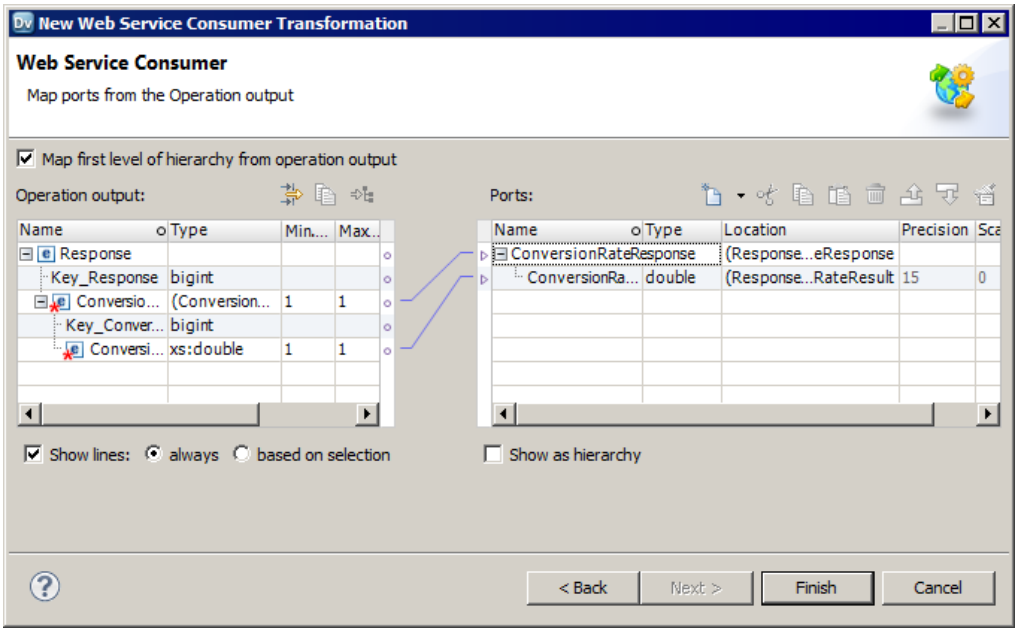
   You can choose an operation that has a SOAP 1.1 binding or SOAP 1.2 binding.
4. Click **Next**.

The **Map Ports to the Operation Input** screen appears. The **Ports** area shows the transformation input groups and input ports. The **Operation Input** area shows the request message hierarchy.



5. Define the input groups and input ports and map the input ports to operation input nodes.
6. Click **Next**.

   The **Map Ports from the Operation Output** screen appears. The **Operation Output** area shows the response message hierarchy. The **Ports** area shows the transformation output groups and output ports.



7. Define the output groups and output ports, and map the operation output nodes to the output ports.
8. Click **Finish**.
9. Click the **Advanced** view to configure the transformation properties and the web services connection.

Web Service Consumer Transformation Example

Your organization needs to expose order information for the RT100 product line to the sales organization. The sales team needs to query the order summary and order details on a daily basis.

Create a logical data object that exposes the daily order information in virtual tables. The read mapping contains a Web Service Consumer transformation that returns the latest RT100 orders. The Web Service Consumer transformation consumes a web service that returns the daily order summary and order detail information for the RT100 product line.

Input File

The input file is a flat file that contains the product line number.

Create a physical data object to define the input file. The file has one field, Product_Line. The field value is RT100. Define the location of the physical data object in the **Runtime Properties** view.

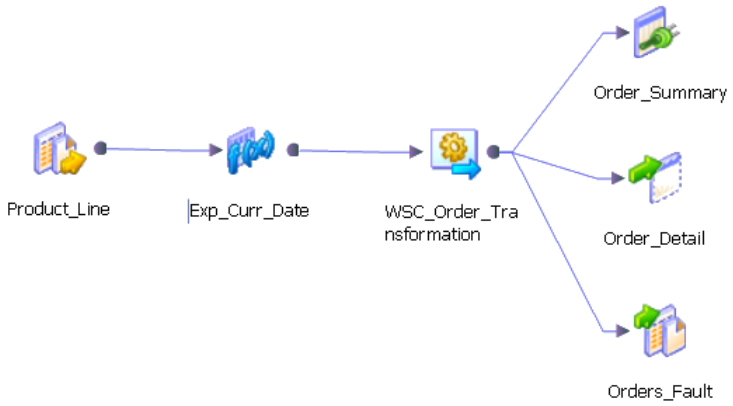
Logical Data Object Model

A business analyst in your organization creates a logical data model that describes the order summary and order detail table structures. The logical data model contains the Order_Summary and Order_Detail logical data objects.

The analyst creates a schema in a modeling tool that defines the logical data model. You import the logical data model from the schema and create the Order_Summary and Order_Detail logical data objects.

Logical Data Object Mapping

The logical data object mapping describes how to access data through the logical data object.



The read mapping contains the following objects:

Product_Line

Input flat file that contains the product line number.

Exp_Curr_Date transformation

Expression transformation that returns the current date and a primary key for the Web Service Consumer transformation root level input group.

WSC_Order transformation

Web Service Consumer transformation that consumes a web service to retrieve order information. The transformation passes the product line and current date to the web service in the request message. The transformation receives order information from the web service in the response message.

Order_Summary table

A logical data object that contains order information such as Order_No, Customer_Id, Qty, and Order_Date.

Order_Detail table

A logical data object that contains order detail information such as Order_No, Product_Id, Qty, and Status.

Orders_Fault

Output flat file that receives generic fault messages.

Web Service Consumer Transformation

The Web Service Consumer transformation receives a product line, date, and a sequence number as input. The transformation consumes the Get_Order_Info web service operation to retrieve the order information.

When you create the Web Service Consumer transformation, choose a WSDL data object that describes the request and response web service messages. A web service message contains hierarchical groups of XML elements. An element can contain other elements. Some elements occur multiple times. Create the transformation from the Order_Info WSDL object in the repository.

Configure the transformation input ports and map the ports to the operation input hierarchy. Map nodes from the operation output hierarchy to the output ports. Define the web services connection and run-time properties.

Transformation Input Mapping

When you show the input mapping on the **Ports** view, you can define input ports and map them to nodes in the operation input.

The transformation **Ports** area has a root group and an Order group. The root group is the Request input group. Add one port to the Request input group to represent the primary key.

The Order group has the **Select_Date** and **Select_Product_Line** input ports.

Map the input ports to the **Order_Date** and **Product_Line** nodes in the **Operation Input** area.

The **Operation Input** area defines the request message that the Web Service Consumer transformation passes to the web service. The nodes appear in the **Operation Input** area by default.

Transformation Output Mapping

When you show the output mapping on the **Ports** view, you can define the output ports by mapping nodes from the operation output to the transformation output groups.

The web service returns the following hierarchy in a web service response message:

```
Response
   Orders
      Order
         Key_Order
         Order_ID
         Order_Date
```

```
Customer_ID
Total_Qty
Order_Details
   Order_Detail
      Product_ID
      Description
      Qty
      Status
```

The web service returns multiple orders. Order is a multiple-occurring node in the Orders level. For each order, the web service can return multiple order details. Order_Detail is a multiple-occurring node in the Order_Details level.

Note: The Developer tool adds the Key_Order node in the user interface. You can map the key to output groups to define relationships between groups. For this example, the Order_ID is the primary key in Order, and it is the foreign key in Order_Details.

Create the following output groups in the **Ports** area:

```
Order
    Order_ID
    Order_Date
    Customer_ID
    Total_Qty

Order_Detail
    Order_ID
    Product_ID
    Description
    Qty
    Status
```

The Data Integration Service writes a row from the Order group whenever the value of Order_ID changes.

The Data Integration Service writes a row from the Order_Detail group whenever the values of Order_ID and Product_ID change.

Transformation Advanced Properties

Configure the following advanced properties for the Web Service Consumer transformation:

Enable Generic SOAP Fault Handling

Adds output ports that receive SOAP fault messages.

Connection

Choose a web services connection to access the web service.

Enable Compression

The Web Service Consumer transformation compresses the web messages with GZIP.

CHAPTER 10

REST Web Service Consumer Transformation

This chapter includes the following topics:

- REST Web Service Consumer Transformation Overview, 113
- REST Web Service Consumer Transformation Configuration, 115
- HTTP Methods, 116
- REST Web Service Consumer Transformation Ports, 119
- REST Web Service Consumer Transformation Input Mapping, 122
- REST Web Service Consumer Transformation Output Mapping, 124
- REST Web Service Consumer Transformation Advanced Properties, 126
- REST Web Service Consumer Transformation Creation, 127

REST Web Service Consumer Transformation Overview

The REST Web Service Consumer Transformation is an active transformation that connects to a REST web service as a web service client to access or transform data. Use a REST Web Service Consumer transformation to connect to a REST web service. The REST Web Service Consumer transformation can send a request to a REST web service and receive a response from a REST web service.

The REST Web Service Consumer transformation connects to a web service through a URL that you define in the transformation or in an HTTP connection. You can also use an HTTPS connection. REST Web Service Consumer transformations can use TLS 1.2, TLS 1.1, or TLS 1.0.

A REST web service contains an HTTP method for each action that the web service supports. When the Data Integration Service connects to a REST web service, it can send a request to get, post, put, or delete data. The request can act upon individual resources or collections of resources. After the Data Integration Service sends a request message, it receives a response message from the web service.

The request and the response messages contain XML or JSON data with elements that can form a hierarchy. When a request or a response message contains multiple-occurring elements, groups of elements form levels in the XML or JSON hierarchy. Groups are related when one level is nested within another.

In the REST Web Service Consumer transformation, the method input and method output define the structure of the request and the response messages. The method input and method output include mappings that define how to map the message elements to the input and output ports.

The REST Web Service Consumer transformation supports proxy server. You can also connect to a Microsoft SharePoint application with the REST Web Service Consumer transformation.

Example

An online store defines resources for a products database. The database identifies each product by part number.

Web service clients access the product details through a REST web service. The web service uses the following URL:

```
http://www.HypoStores.com/products/ProductDetails
```

You need to retrieve details about a specific product, such as the description and the unit price, and pass the details to a transformation downstream in a mapping. Create a REST Web Service Consumer transformation to retrieve details about a product and pass them to another transformation..

The following table shows the transformation details that you configure:

Transformation Detail	Value
HTTP method	Get
Base URL	`http://www.HypoStores.com/products/ProductDetails`
Input argument port	Part_No
Output ports	Description, Unit_Price
Method output	<The structure of the response message.>

The method output includes an output mapping that defines how the elements in the response message map to the output ports.

When the Data Integration Service sends the request to the web service, it appends the value in the argument port to the base URL. For example, to retrieve details about part 0716, the Data Integration Service uses the following URL:

```
http://www.HypoStores.com/products/ProductDetails?Part_No=0716
```

When the Data Integration Service receives a response, it converts the product description and the unit price in the response message to data for the output ports.

You can also pass Part_No as a parameter and substitute the value mid-stream when you run the mapping.

REST Web Service Consumer Transformation Process

The REST Web Service Consumer transformation creates a request message based on the data in the input ports and the method input. It converts elements in the response message to data for the output ports based on the method output.

The input ports of the REST Web Service Consumer transformation contain relational data from upstream transformations in a mapping. The Data Integration Service uses the method input to convert data from the input ports to elements in the request message.

To connect to the web service, the Data Integration Service reads the base URL that you configure in the transformation properties or the HTTP connection. It identifies the resource that you want to get, post, put, or delete by appending values from the URL ports or argument ports to the base URL.

When the Data Integration Service receives a response, it passes the data in the response message to the output ports of the transformation. The Data Integration Service passes data based on how you configure the method output. The output ports contain relational data. The Data Integration Service sends the data in the output ports to downstream transformations in the mapping or to the target.

REST Web Service Consumer Transformation Configuration

When you create a REST Web Service Consumer transformation, you select the HTTP method and define the method input and output. If you select the Get method, you do not define method input.

The input elements in the HTTP request message map to input ports. The output elements in the HTTP response message map to output ports. The Developer tool creates ports for the first level elements.

When you configure the transformation, you complete the following tasks:

1. Select the HTTP method.
2. Configure ports to represent elements in the header and body of the request and response messages.
3. Configure the input mapping.
4. Configure the output mapping.
5. Configure advanced properties such as the connection and the base URL for the web service.

If the REST web service requires authentication, create an HTTP connection object.

Message Configuration

The Data Integration Service generates request messages and interprets response messages based on the method input and output and on the ports that you configure in the REST Web Service Consumer transformation.

Input ports represent different parts of the request message. You can add input ports that identify the resource that you want to retrieve or change. You can also add input ports that represent HTTP headers, cookie information, and elements in the request message.

Output ports represent elements in the response message that you want to send to downstream transformations or to the target in a mapping. You can add output ports that represent HTTP headers, cookie information, the response code, and elements in the response message.

Resource Identification

To identify the resource in an HTTP request, the Data Integration Service appends values in specific input ports to the base URL. You define the base URL in the HTTP connection or in the transformation properties. Use URL or argument ports to identify a particular resource.

Use URL ports when the web service identifies a resource through a unique string of characters.

For example, the HypoStores REST web service identifies parts by the part number through the following URL:

```
http://www.HypoStores.com/products/ProductDetails/<Part_No>
```

To identify a part, define the following transformation details:

1. Set the base URL to the following URL:

   `http://www.HypoStores.com/products/ProductDetails`

2. Define a URL port, and pass the part number to the transformation through the URL port.

If the mapping passes part number 500 to the URL port, the Data Integration Service uses the following URL in the request message:

```
http://www.HypoStores.com/products/ProductDetails/500
```

Use argument ports when the web service identifies the location of a resource through arguments.

For example, you want to pass a part number to the HypoStores REST web service through the "Part_No" argument.

To identify a part, define the following transformation details:

1. Set the base URL to the following URL:

   `http://www.HypoStores.com/products/ProductDetails`

2. Create an argument port with argument name "Part_No," and pass the part number to the transformation through the argument port.

If the mapping passes part number 600 to the argument port, the Data Integration Service uses the following URL in the request message:

```
http://www.HypoStores.com/products/ProductDetails?Part_No=600
```

Create multiple argument ports to define multiple arguments. The Data Integration Service separates each argument with an ampersand character (&).

For example, you want to retrieve employee details from a REST web service and pass the employee first name and last name through the "First_Name" and "Last_Name" arguments. Create argument ports with the argument names "First_Name" and "Last_Name." If the mapping passes the name "John Smith" to the transformation, the Data Integration Service uses a URL like the following in the request message:

```
http://www.HypoStores.com/employees/EmpDetails?First_Name=John&Last_Name=Smith
```

If you do not specify a URL or an argument port, the Data Integration Service uses the base URL from the transformation properties or HTTP connection to identify the resource. The base URL in the HTTP connection overrides the base URL in the transformation.

HTTP Methods

When you create a REST Web Service Consumer transformation, you select the HTTP method that the Data Integration Service uses in the request message. You cannot change the HTTP method after you create the transformation.

You configure the transformation to use one of the following HTTP methods:

Get

Retrieves a resource or a collection of resources from the web service. For example, you can retrieve a table of products or retrieve information about one product.

Post

Sends data to a web service. Use the Post method to create a resource or collection of resources. For example, you can add the details of a new store transaction.

Put

Replaces a resource or a collection of resources. If the data does not exist, the Put method posts the data. For example, you can update a customer shipping address.

Delete

Deletes a resource or a collection of resources. For example, you can delete the record of an employee that no longer works for an organization.

HTTP Get Method

The Data Integration Service uses the HTTP Get method to retrieve data from a REST web service. Use the Get method to retrieve a resource or a collection of resources.

When you configure the REST Web Service Consumer transformation to use the Get method, you configure the input ports, the method output, and the output ports. You do not configure method input.

Example

You want to retrieve the description and price for part number 500 in the HypoStores products database. The web service uses the following URL to identify a part:

```
http://www.HypoStores.com/products/ProductDetails?Part_No=<Part_No>
```

Enter the following base URL:

```
http://www.HypoStores.com/products/ProductDetails
```

The following table shows the input port that you might define:

Port Type	Argument Name	Input Value
Argument	Part_No	500

The following table shows the output ports that you might define:

Port Type	Port Name	Return Value
Output	Part_Desc	…<desc>ACME ball point pens, 12-pk, black, 0.7 mm</desc>…
Output	Price_USD	…<price>9.89</price>…

HTTP Post Method

The Data Integration Service uses the HTTP Post method to send data to a REST web service. The web service determines the actual function that the Post method performs. You might use the Post method to create a resource or a collection of resources.

When you configure the REST Web Service Consumer transformation to use the Post method, you configure the input ports, the method input, the method output, and the output ports.

Example

You want to post new part 501 to the HypoStores products database. The web service uses the following URL for part 501:

```
http://www.HypoStores.com/products/ProductDetails/501
```

Enter the following base URL:

```
http://www.HypoStores.com/products/ProductDetails
```

The following table shows the input ports that you might define:

Port Type	Port Name	Input Value
URL	URL_Part_No	501
Input	Part_Desc	ACME ball point pens, 12-pk, black, 0.5 mm
Input	Price_USD	9.89

The following table shows the output ports that you might define:

Port Type	Port Name	Return Value
Output	Response	<Response returned by the web service>

HTTP Put Method

The Data Integration Service uses the HTTP Put method to update data through a REST web service. Use the Post method to update a resource or a collection of resources. If the data does not exist, the Data Integration Service creates the resource or collection of resources.

When you configure the REST Web Service Consumer transformation to use the Put method, you configure the input ports, the method input, the method output, and the output ports.

Example

You want to update the unit price for part 501 in the HypoStores products database. The web service uses the following URL for part 501:

```
http://www.HypoStores.com/products/ProductDetails/501
```

Enter the following base URL:

```
http://www.HypoStores.com/products/ProductDetails
```

The following table shows the input ports that you might define:

Port Type	Port Name	Input Value
URL	URL_Part_No	501
Input	Price_USD	9.99

The following table shows the output ports that you might define:

Port Type	Port Name	Return Value
Output	Response	<Response returned by the web service>

HTTP Delete Method

The Data Integration Service uses the HTTP Delete method to remove data through a REST web service. Use the Delete method to remove a resource or a collection of resources.

When you configure the REST Web Service Consumer transformation to use the Delete method, you configure the input ports, the method input, the method output, and the output ports.

Example

You want to delete part number 502 from the HypoStores products database. The web service uses the following URL to identify a part:

```
http://www.HypoStores.com/products/ProductDetails?Part_No=<Part_No>
```

Enter the following base URL:

```
http://www.HypoStores.com/products/ProductDetails
```

The following table shows the input port that you might define:

Port Type	Argument Name	Input Value
Argument	Part_No	502

The following table shows output ports that you might define:

Port Type	Port Name	Return Value
Output	Response	<Response returned by the web service>

REST Web Service Consumer Transformation Ports

A REST Web Service Consumer transformation can have multiple input ports and multiple output ports. You create ports in groups based on the structure of the XML or JSON hierarchy.

When you view the transformation ports, show the ports if you do not need to view the XML or JSON hierarchy. When you show the ports, you can define groups, define ports, and map elements from the method input and output to input and output ports.

A REST Web Service Consumer transformation can have multiple input groups and multiple output groups. When you create ports, create groups and add the ports to the groups. Define the ports in a group hierarchy based on the structure of the input or output hierarchy in the XML or JSON . Add a key to relate a child group to a parent group.

All groups except the lowest group in the hierarchy must have primary keys. All groups in the hierarchy except the root group must have foreign keys.

The transformation has a root input group named RequestInput. You must add a primary key to the root input group. The key must be string, bigint, or integer. You can configure any port in the root input group as a pass-through port.

To map an element to a port, click the field in the **Location** column and expand the hierarchy in the **Select Location** dialog box. Then, choose an element from the hierarchy.

Input Ports

Input ports represent data from an upstream transformation or source that you want to pass to the web service. You can configure multiple input ports. Each input port maps to an element in the request message.

To add an input port, select an input group, click the arrow next to the **New** button, and select **Field**.

Output Ports

Output ports represent elements in the response message that you want to pass to a downstream transformation or to the target. You can configure multiple output ports. Each output port maps to an element in the response message.

To add an output port, select an output group, click the arrow next to the **New** button, and select **Field**.

Pass-through Ports

Pass-through ports pass data through the transformation without changing the data. You can configure any port in the root input group as a pass-through port.

To add a pass-through port, add a port to the root input group. Then right-click the port and select **Map**.

Argument Ports

Argument ports allow you to identify a resource when the URL for the resource takes an argument. Add argument ports to the root input group.

An argument port has a port name and an argument name. If an argument name contains a character that is not allowed in a port name, enter an argument name that differs from the port name. For example, you want to pass the argument "Cust-ID" to the web service, but the Data Integration Service does not allow the dash character (-) in port names. Enter "Cust-ID" as the argument name, but enter "CustID" as the port name.

The Data Integration Service appends the argument names and values for each argument port to the base URL as name=value pairs. You can configure multiple argument ports. The Data Integration Service separates multiple arguments in the request with an ampersand character (&).

For example:

```
http://www.HypoStores.com/customers/CustDetails?Last_Name=Jones&First_Name=Mary
```

If you define argument ports and URL ports in the transformation, the Data Integration Service appends the URL port values to the base URL, followed by the argument names and values.

To add an argument port, right-click on the root input group and select **New** > **Argument Ports**. Enter the argument name and the port name.

URL Ports

URL ports allow you to identify a resource through a static URL. To identify a resource, the Data Integration Service appends the value of the URL port to the base URL.

For example:

```
http://www.HypoStores.com/products/ProductDetails/<URL_port_value>
```

Add URL ports to the root input group.

You can configure multiple URL ports. The Data Integration Service separates the values in each URL port with a slash character (/). If you define URL ports and argument ports in the transformation, the Data

Integration Service appends the URL port values to the base URL, followed by the argument names and values.

To add a URL port, right-click on the root input group and select **New** > **URL Ports**.

HTTP Header Ports

HTTP header ports represent HTTP headers in the request or the response message. You can configure multiple HTTP header ports.

To pass header information to the web service in the request, add the port to the root input group. You can configure one HTTP header port for the root input group. If you add an HTTP header to the root input group, you can configure it as a pass-through port.

To extract header information from the response, add the port to an output group. You can configure one HTTP header port for each output group.

An HTTP header port has a port name and an HTTP header name. If an HTTP header name contains a character that is not allowed in a port name, enter an HTTP header name that differs from the port name. For example, you want to pass the header name "Content-Type" to the web service, but the Data Integration Service does not allow the dash character (-) in port names. Enter "Content-Type" as the HTTP header name, but enter "ContentType" as the port name.

To add an HTTP header port, right-click on the root input group and select **New** > **HTTP Header**. Enter a header name and port name.

Cookie Ports

You can configure the REST Web Service Consumer transformation to use cookie authentication. The remote web server tracks the web service consumer users based on the cookies. You can increase performance when a mapping calls a web service multiple times.

To pass cookie information to the web service in the request, add the port to the root input group. You can configure one cookie port for the root input group. If you add a cookie port to the root input group, you can configure it as a pass-through port.

To extract cookie information from the response, add a cookie port to an output group. You can configure one cookie port for each output group.

When you project the cookie port to a web service request message, the web service provider returns a cookie value in the response message. You can pass the cookie value to another transformation downstream in the mapping, or you can save the cookie value in a file. When you save the cookie value in a file, you can configure the cookie as input to the REST Web Service Consumer transformation.

To add a cookie port, right-click on the root input group and select **New** > **Other Ports**. Then, select **Cookie**, and click **OK**.

Output XML Ports

Output XML ports represent responses from the web service. Output XML ports are string ports.

Add an output XML port to an output group. You can configure one output XML port for each output group.

right-click on the root input group and select **New** > **Other Ports**. Then, select **Output XML**, and click **OK**.

Response Code Ports

Response code ports represent the HTTP response codes from the web service. Response code ports are integer ports.

Add a response code port to an output group. You can configure one response code port for each output group.

To add a response code port, select an output group, right-click on the root input group and select **New** > **Other Ports**. Then, select **Response Code**, and click **OK**. .

REST Web Service Consumer Transformation Input Mapping

When you view the transformation ports, show the input mapping to view the method input hierarchy. When you show the input mapping, you can define input groups, define input ports, and map input ports to method input elements.

The input mapping includes the following areas:

Ports

Create the transformation input groups and input ports in the **Ports** area.

Method Input

The **Method Input** area shows the elements in the request message that the REST Web Service Consumer transformation sends to the web service. If you use a schema object to create the transformation, the schema object defines the method input hierarchy.

After you create input ports, map the input ports from the **Ports** area to the elements in the **Method Input** area. When you map an input port to an element in the method input, the location of the port appears in the Location column in the **Method Input** area.

The Developer tool maps elements in the first level of the method input to input ports when you choose to map the first level of the input hierarchy. The Developer tool also creates the ports to perform the mapping. If the first level of the hierarchy contains a multiple occurring parent element with one or more multiple occurring child elements, the Developer tool does not map the first level of the hierarchy.

You can choose to view the lines that connect the input ports to the elements in the method input.

Rules and Guidelines to Map Input Ports to Elements

Review the following rules when you map input ports to elements in the method input hierarchy:

- You can map an input port to one element in the hierarchy. You can map the same port to any number of keys in the hierarchy.
- The input port and the element must have compatible datatypes.
- You can map ports from one input group to multiple hierarchy levels in the method input.
- You must map input ports to the keys in the method input. Any port that you map to a key must be of string, integer, or bigint datatype. Map data to the keys in all levels in the method input above the hierarchy level that you are including in the request message. Include the foreign keys for all levels above and including the level you are mapping.

Note: You do not have to map input ports to keys if you are mapping only the lowest level of the method input hierarchy.

- You must map the RequestInput root element to the child element of Rest_Consumer_input group for method input definition.
- You can map multiple string, bigint, or integer input ports to a key in the **Method Input** area to create a composite key. When you click the **Location** field for a composite key, you can reorder the input ports or remove one of the ports.
- If the web service produces a JSON document, ensure that xmlRoot is the first node in the response hierarchy. If xmlRoot is not the first node for a web service with JSON response, null values may appear.

Mapping Input Ports to the Method Input

When you show the transformation input mapping, you can define input groups, define input ports, and map input ports to method input elements.

1. Open a REST Web Service Consumer transformation.
2. In the **Ports** view, show the input mapping.
3. Define a primary key for the root input group.
4. To add an input group or port to the **Ports** area, use one of the following methods:

Method	Description
Drag an element.	Drag a group or a child element from the **Method Input** area to an empty column in the **Ports** area. If you drag a group to the **Ports** area, the Developer tool adds a group without ports.
Manually add a group or port.	To add a group, click the arrow next to the **New** button, and then click **Group**. To add a port, click the arrow next to the **New** button, and then click **Field**.
Drag a port from another transformation.	In the editor, drag a port from another transformation to the REST Web Service Consumer transformation.
Copy a port.	Select ports from another transformation and copy them to the **Ports** area. To copy ports, you can use keyboard shortcuts or you can use the **Copy** and **Paste** buttons in the Developer tool.
Select **Map first level of hierarchy**.	The Developer tool maps elements in the first level of the method input to input ports and groups. The Developer tool also creates the input ports and groups to perform the mapping.

5. If you manually create a port or you copy a port from another transformation, click the **Location** column in the **Method Input** area and choose a port from the list.
6. To map input ports as a composite key, use one of the following methods:

Method	Description
Drag input ports.	Select two or more input ports, and drag them to a key in the method input hierarchy.
Select input ports from the **Select Location** dialog box.	Click the **Location** column of a key in the method input hierarchy and then select the input ports.

7. To clear the locations of elements, use one of the following methods:

Method	Description
Click **Clear**.	Select one or more elements in the **Method Input** area and click **Clear**.
Delete the lines that connect ports to elements.	Select one or more lines that connect the input ports to the elements in the method input, and press **Delete**.

REST Web Service Consumer Transformation Output Mapping

When you view the transformation ports, show the output mapping to view the method output hierarchy. When you show the output mapping, you can define output groups, define output ports, and map method output elements to output ports.

The output mapping includes the following areas:

Method Output

The **Method Output** area shows the elements in the response message that the web service returns to the REST Web Service Consumer transformation. If you use a schema object to create the transformation, the schema object defines the method output hierarchy.

Ports

Create the transformation output groups and ports in the **Ports** area.

After you create output ports, map the elements from the **Method Output** area to the ports in the **Ports** area. When you map an element from the method output to an output port, the location of the element appears in the **Location** column in the **Ports** area.

The Developer tool maps elements in the first level of the method output to output ports when you choose to map the first level of the output hierarchy. The Developer tool also creates the ports to perform the mapping. If the first level of the hierarchy contains a multiple occurring parent element with one or more multiple occurring child elements, the Developer tool does not map the first level of the hierarchy.

You can choose to display the output ports in a hierarchy. Each child group appears under the parent group. You can also choose to view the lines that connect the elements in the method output to the output ports.

If the associated schema object is deleted from the repository, the Developer tool retains the location of the method elements in the output mapping. When you show the output mapping, the **Ports** area still displays the location of the method elements in the **Location** column for the output ports. If you associate another schema with the transformation, the Developer tool checks whether each location is valid. The Developer tool clears the location of method elements in the **Ports** area of the output mapping if the location is no longer valid.

Rules and Guidelines to Map Elements to Output Ports

Review the following rules when you map elements in the method output hierarchy to output ports:

- The method output element and the output port must have compatible datatypes.
- You cannot map an element to more than one output port in a group.

- Each output port must have a valid location, unless the port is a pass-through port.
- If you drag a multiple-occurring child element to an empty output port, you must relate the group to other output groups. When you select a group, the Developer tool creates keys to relate the groups.
- When you drag a multiple-occurring element into a group that contains the parent element, you can configure the number of child element occurrences to include. Or, you can replace the parent group with the multiple-occurring child group in the transformation output.
- If the web service produces a JSON document, ensure that xmlRoot is the first node in the response hierarchy. If xmlRoot is not the first node for a web service with JSON response, null values may appear in the output ports.

Customize View Options

You can change the method output hierarchy to show the cookie ports, pass-through ports, and keys in the **Method Output** area. You can also show grouping constructs that define how to order elements.

Click **Customize View** in the **Method Output** area. Enable any of the following options:

Sequence, Choice, and All

Show a line that indicates whether an element definition is sequence, choice, or all.

Elements in a sequence group must be in the order specified in the hierarchy.

At least one element in a choice group must appear in the response message.

Elements in an all group must all be included in the response message.

Keys

View the keys in the **Method Output** area. The **Method Output** area includes keys for each group. You can add a key to an output port in the **Ports** area.

Pass-through Ports

The **Method Output** area shows the pass-through ports. Pass-through ports are ports that pass data through the transformation without changing the data. You can project pass-through ports from the method output to any of the REST Web Service Consumer transformation output groups. A pass-through port receives data one time, so the port is at the root level in the response messages.

Mapping the Method Output to Output Ports

When you show the transformation output mapping, you can define output groups, define output ports, and map method output elements to output ports.

1. Open a REST Web Service Consumer transformation.
2. In the **Ports** view, show the output mapping.
3. To add an output group or port to the **Ports** area, use one of the following methods:

Method	Description
Drag an element.	Drag a group or a child element in the **Method Output** area to an empty column in the **Ports** area. If you drag a group to the **Ports** area, the Developer tool adds a group without ports.

Method	Description
Manually add a group or port.	To add a group, click the arrow next to the **New** button and then click **Group**. To add a port, click the arrow next to the **New** button, and then click **Field**.
Drag a port from another transformation.	In the editor, drag a port from another transformation to the REST Web Service Consumer transformation.
Copy a port.	Select ports from another transformation and copy them to the **Ports** area. To copy ports, you can use keyboard shortcuts or you can use the **Copy** and **Paste** buttons in the Developer tool.

4. If you manually create a port or you copy a port from another transformation, click the **Location** column in the **Ports** area, and choose an element from the list.
5. To clear the locations of ports, use one of the following methods:

Method	Description
Click **Clear**.	Select one or more ports in the **Ports** area, and click **Clear**.
Delete the lines that connect elements to ports.	Select one or more lines that connect the elements in the method output to the output ports, and press **Delete**.

REST Web Service Consumer Transformation Advanced Properties

Configure properties that help determine how the Data Integration Service processes data for the REST Web Service Consumer transformation.

Configure the following properties on the **Advanced** tab:

Tracing Level

Amount of detail that appears in the log for this transformation. You can choose terse, normal, verbose initialization, or verbose data. Default is normal.

Connection

Identifies the HTTP connection object to connect to the web service. Create and edit the HTTP connection in the Developer tool. When you configure an HTTP connection, configure the base URL, the type of security that the web service requires, and a connection timeout period.

The REST Web Service Consumer transformation connects to a web service using a URL. You can define the URL in the transformation properties or in the HTTP connection.

Configure an HTTP connection in the following circumstances:

- You do not use a URL input port.
- The web service requires HTTP authentication or SSL certificates.
- You want to change the default connection timeout period.

XML Schema Validation

Validates the response message at run time. **Select Error on Invalid XML** or **No Validation**.

Sorted Input

Enables the Data Integration Service to generate output without processing all of the input data. Enable sorted input when the input data is sorted by the keys in the XML input hierarchy.

URL

The base URL for the REST web service. The base URL in the HTTP connection overrides this value.

Format

The format of the web service response. Select **XML** or **JSON** depending on the web service response..

REST Web Service Consumer Transformation Creation

You can create a reusable or non-reusable REST Web Service Consumer transformation. Reusable transformations can exist in multiple mappings. Non-reusable transformations exist within a single mapping.

When you create a REST Web Service Consumer transformation, you can define the elements and XML hierarchy manually or you can import the elements and hierarchy from a schema object. The schema object can be an XML file or a text file.

Creating a REST Web Service Consumer Transformation

When you create a REST Web Service Consumer transformation, you select a method and define the method input and the method output based on the method that you choose.

1. To create a REST Web Service Consumer transformation, use one of the following methods:

Method	Description
Reusable	Select a project or folder in the Object Explorer view. Click **File** > **New** > **Transformation**. Select the REST Web Service Consumer transformation and click **Next**.
Non-reusable	In a mapping or mapplet, drag a REST Web Service Consumer transformation from the transformation palette to the mapping or mapplet editor.

2. Enter the transformation name, and select the location and the HTTP method.
3. Click **Next**.
4. To define the method input, use one of the following methods:

Method	Description
Create as empty	Define the XML elements and hierarchy manually.
Create from an element in a Schema Object	Import the XML elements and hierarchy from a schema object.

The **Method input definition** area shows the transformation input groups and input ports. The **Input mapping** area shows the request message hierarchy.

5. Define the input groups and input ports, and map the input ports to input elements.
6. Click **Next**.

7. To define the method output, select **Create as empty** or **Create from an element in a Schema Object**.

   The **Method output definition** area shows the transformation input groups and input ports. The **Output mapping** area shows the request message hierarchy.

8. Define the output groups and output ports, and map the elements to output ports.
9. Click **Finish**.

APPENDIX A

Datatype Compatibility

This appendix includes the following topics:

- Datatype Reference Overview, 129
- XML and Transformation Datatypes, 129
- Decimal, 131

Datatype Reference Overview

When you create a mapping, you create a set of instructions for the Data Integration Service to read data from a source, transform it, and write it to a target. The Data Integration Service transforms data based on dataflow in the mapping, starting at the first transformation in the mapping, and the datatype assigned to each port in a mapping.

The Developer tool displays two types of datatypes:

- Native datatypes. Specific to the relational table or flat file used as a physical data object. Native datatypes appear in the physical data object column properties.
- Transformation datatypes. Set of datatypes that appear in the transformations. They are internal datatypes based on ANSI SQL-92 generic datatypes, which the Data Integration Service uses to move data across platforms. The transformation datatypes appear in all transformations in a mapping.

When the Data Integration Service reads source data, it converts the native datatypes to the comparable transformation datatypes before transforming the data. When the Data Integration Service writes to a target, it converts the transformation datatypes to the comparable native datatypes.

When you specify a multibyte character set, the datatypes allocate additional space in the database to store characters of up to three bytes.

XML and Transformation Datatypes

XML datatypes map to transformation datatypes that the Data Integration Service uses to move data across platforms.

The Data Integration Service supports all XML datatypes specified in the W3C May 2, 2001 Recommendation. However, the Data Integration Service may not support the entire XML value range. For more information about XML datatypes, see the W3C specifications for XML datatypes at the following location: http://www.w3.org/TR/xmlschema-2.

The following table compares XML datatypes to transformation datatypes:

Datatype	Transformation	Range
anyURI	String	1 to 104,857,600 characters
base64Binary	Binary	1 to 104,857,600 bytes
boolean	String	1 to 104,857,600 characters
byte	Integer	-2,147,483,648 to 2,147,483,647
date	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
dateTime	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
decimal	Decimal	Precision 1 to 28, scale 0 to 28
double	Double	Precision of 15 digits
duration	String	1 to 104,857,600 characters
ENTITIES	String	1 to 104,857,600 characters
ENTITY	String	1 to 104,857,600 characters
float	Double	Precision of 15 digits
gDay	String	1 to 104,857,600 characters
gMonth	String	1 to 104,857,600 characters
gMonthDay	String	1 to 104,857,600 characters
gYear	String	1 to 104,857,600 characters
gYearMonth	String	1 to 104,857,600 characters
hexBinary	Binary	1 to 104,857,600 bytes
ID	String	1 to 104,857,600 characters
IDREF	String	1 to 104,857,600 characters
IDREFS	String	1 to 104,857,600 characters
int	Integer	-2,147,483,648 to 2,147,483,647
integer	Integer	-2,147,483,648 to 2,147,483,647
language	String	1 to 104,857,600 characters
long	Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
Name	String	1 to 104,857,600 characters
NCName	String	1 to 104,857,600 characters

Datatype	Transformation	Range
negativeInteger	Integer	-2,147,483,648 to 2,147,483,647
NMTOKEN	String	1 to 104,857,600 characters
NMTOKENS	String	1 to 104,857,600 characters
nonNegativeInteger	Integer	-2,147,483,648 to 2,147,483,647
nonPositiveInteger	Integer	-2,147,483,648 to 2,147,483,647
normalizedString	String	1 to 104,857,600 characters
NOTATION	String	1 to 104,857,600 characters
positiveInteger	Integer	-2,147,483,648 to 2,147,483,647
QName	String	1 to 104,857,600 characters
short	Integer	-2,147,483,648 to 2,147,483,647
string	String	1 to 104,857,600 characters
time	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
token	String	1 to 104,857,600 characters
unsignedByte	Integer	-2,147,483,648 to 2,147,483,647
unsignedInt	Integer	-2,147,483,648 to 2,147,483,647
unsignedLong	Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
unsignedShort	Integer	-2,147,483,648 to 2,147,483,647

Decimal

When a web service operation mapping contains an Input or Output transformation with a Decimal data type of precision greater than 28 digits, the Data Integration Service converts the Decimal data type to the Double data type.

INDEX

A

abstract property
  schema object 23
advanced properties
  REST Web Service Consumer transformation 126
  Web Service Consumer transformation 102
Advanced tab
  Output transformation 50
all group
  viewing in REST Web Service Consumer transformation 125
  viewing in Web Service Consumer transformation 97, 100
any elements
  Web Service Consumer transformation 96, 99
anyAttribute attributes
  Web Service Consumer transformation 96, 99
anyType
  map ports 77
anyType elements
  parsing 64
  Web Service Consumer transformation 96, 99
attribute properties
  schema object 26
attributeFormDefault
  schema object 27
authentication
  UsernameToken 85

B

base property
  schema object 25
binding
  WSDL file element 92
block property
  schema object 24

C

certificates
  adding untrusted certificates 20, 32
  certificate properties 20, 32
  managing certificates 20, 31
  untrusted certificates 20, 31
choice elements
  description 78
  parsing SOAP messages 67
  viewing in REST Web Service Consumer transformation 125
  viewing in Web Service Consumer transformation 97, 100
code
  web service fault handling 55
collapse whitespace property
  schema object 25
complex type
  advanced properties 26
composite keys
  REST Web Service Consumer transformation 122
  Web Service Consumer transformation 96
concurrent web service request messages
  enabling in Web Service Consumer transformation 102
connection
  REST web services 126
  web services 102
cookie authentication
  REST Web Service Consumer transformation 121
  Web Service Consumer transformation 95
creating
  fault transformation 53
customize view options
  description 59

D

datatypes
  overview 129
  XML 129
denormalized input
  web service ports 75
denormalized output
  SOAP message parsing 63
derived type elements
  Web Service Consumer transformation 96, 99
derived types
  parsing SOAP messages 65
  web services 77
detail element
  web service fault handling 55
dynamic URL
  Web Service Consumer transformation 95

E

early selection optimization
  Web Service Consumer transformation 106
elementFormDefault
  schema object 27
elements
  creating 43
  union 79
endpoint URL
  REST Web Service Consumer transformation 120
  Web Service Consumer transformation 95
enumeration property
  schema object 23

F

fault actor
  web service fault handling 55
fault code
  web service fault handling 55
fault elements
  SOAP 1.1 55
  SOAP 1.2 55
fault string
  web service fault handling 55
fault transformation
  creating 53
  generic fault 53
  predefined fault 53
Fault transformation
  configuring in a web service 54
  customize view options 59
  description 52
  map the first level of the hierarchy 53
  map to output as XML 52
  mapping rules and guidelines 53
  Ports tab 53
filter port
  Web Service Consumer transformation 106
fixed value property
  schema object 23

G

generated keys
  web service output groups 62
generated prefix
  changing for namespace 22
generic fault
  web service fault handling 58
generic fault output
  enabling in Web Service Consumer transformation 102
generic SOAP faults
  Web Service Consumer transformation 104
GZip
  compressing SOAP messages 104

H

header
  description 13
  HTTP POST 44
headers
  creating 44
HTTP connection
  REST web services 126
HTTP error output
  enabling in Web Service Consumer transformation 102
HTTP header
  adding to REST Web Service Consumer transformation 121
  adding to Web Service Consumer transformation 95
HTTP POST
  header 44
HTTP response code
  REST Web Service Consumer transformation 122

I

inherit by property
  schema object 26
inherit from property
  schema object 26
input mapping
  REST Web Service Consumer transformation 122
  Web Service Consumer transformation 96
Input Ports area
  generating SOAP messages 69
Input transformation
  configuring in a web service 48
  customize view options 59
  description 47
  map the first level of the hierarchy 47
  map to input as XML 47
  mapping rules and guidelines 47
  Ports tab 47

K

keys
  SOAP message hierarchy 71

L

list elements
  description 79
  parsing SOAP messages 67
Location column
  web service transformation 70

M

map the first level of the hierarchy
  Fault Transformation 53
  Input Transformation 47
  Output Transformation 49
map to input as XML
  Input transformation 47
map to output as XML
  Fault transformation 52
  Output transformation 49
maximum length
  schema object 23
maximum occurs
  schema object 23
member types
  schema object 25
minimum length
  schema object 23
minimum occurs
  schema object 23

N

namespace
  changing generated prefix 22
namespaces
  schema object 22
nillible property
  schema object 23

node
web service fault handling 55

O

operation
WSDL file element 92
operation area
web service transformations 70
operation fault
description 13
operation input
description 13
Operation Input area
customizing Web Service Consumer transformation 97
operation mapping
Fault transformation 52
Input transformation 47
Output transformation 49
overview 45
testing 59
operation output
description 13
Operation Output area
customizing Web Service Consumer transformation 100
operations
description 13
output mapping
REST Web Service Consumer transformation 124
Web Service Consumer transformation 99
Output transformation
Advanced tab 50
configuring in a web service 50
customize view options 59
description 49
map the first level of the hierarchy 49
map to output as XML 49
mapping rules and guidelines 50
Ports tab 49
override SOAP action
Web Service Consumer transformation 102

P

password
hashed 86
plain text 86
pattern property
schema object 23
pivoted data
SOAP messages 74
pivoted output
SOAP message parsing 64
ports
denormalized web service input 75
map to SOAP messages 72
Ports tab
Fault transformation 53
Input transformation 47
Output transformation 49
predefined faults
creating 44
web service fault handling 57
push-into optimization
Web Service Consumer transformation 106

Q

Qname elements
parsing SOAP messages 66

R

reason
web service fault handling 55
response code
REST Web Service Consumer transformation 122
REST Web Service Consumer transformation
advanced properties 126
argument ports 120
configuration 115
connection property 126
cookie ports 121
creating 127
customizing output mapping view 125
Delete method 119
Get method 117
HTTP header ports 121
HTTP methods 116
input mapping 122
input mapping rules 122
input ports 120
internet media type 126
mapping elements to ports 119
mapping input 113
mapping input ports 123
mapping output 113
mapping output ports 125
message configuration 115
non-reusable 127
output mapping 124
output mapping rules 124
output ports 120
output XML ports 121
overview 113
pass-through ports 120
ports 119
Post method 117
process 114
proxy server support 113
Put method 118
RequestInput port 119
resource identification 115
response code ports 122
reusable 127
security 93
setting base URL 126
sorted input 126
tracing level 126
transport layer security 93
URL ports 120
XML schema validation 126
result set caching
configuring 83
web service operation properties 83
role
web service fault handling 55

S

schema files
adding to schema objects 22

schema files *(continued)*
editing 30
removing from schema objects 22
setting a default editor 30
schema object
elements advanced properties 24
abstract property 23
attribute properties 26
attributeFormDefault 27
block property 24
complex elements 26
complex elements advanced properties 26
editing a schema file 28
element properties 23
elementFormDefault 27
file location 27
importing 28
inherit by property 26
inherit from property 26
namespaces 22
overview 21
Overview view 21
schema files 22
setting a default editor 30
simple type 25
substitution group 24
synchronization 28
Schema object
Schema view 22
Schema view
Schema object 22
simple type advanced properties 25
security
UsernameToken 85
web service security 84
sequence group
viewing in REST Web Service Consumer transformation 125
service
WSDL file element 92
side effects
Web Service Consumer transformation 106
simple type
schema object 25
SImple Type view
schema object 25
SOAP 1.1
fault elements 55
SOAP 1.2
fault elements 55
SOAP action
overriding in Web Service Consumer transformation 102
SOAP compression
Web Service Consumer transformation 104
SOAP hierarchy
relationship to input ports 70
SOAP message
keys 71
SOAP message parsing
denormalized output 63
derived types 65
description 60
normalized output 62
pivoted output 64
Qname elements 66
union element 67
SOAP messages
map choice elements 78
map list elements 79
SOAP messages *(continued)*
map multiple input ports 74
map multiple-occurring nodes 62
map ports 72
map ports to union elements 79
overview 92
parsing anyType elements 64
parsing choice elements 67
parsing list elements 67
parsing substitution groups 66
pivoting data 74
SOAP web service
development 15
substitution group
schema object 24
substitution groups
parsing SOAP messages 66
Web Service Consumer transformation 96, 99
web services 78
system-defined fault
web service fault handling 57

T

trace level
web service 88
transport layer security
REST Web Service Consumer transformation 93
Web Service Consumer transformation 93
treat fault as error
enabling in Web Service Consumer transformation 102

U

union element
parsing SOAP messages 67
union elements
description 79
UsernameToken
hashed password 86
password digest 87
password types 85
plain text password 86

V

variety property
schema object 25

W

web service
components 13
configuring a Fault transformation 54
configuring an Input transformation 48
configuring an Output transformation 50
configuring result set caching 83
creating a header 44
creating elements 43
creating from WSDL 37
creating predefined faults 44
creating without a WSDL 38
creation 33
deployment 15

web service *(continued)*
derived types 77
examples 15
fault handling 55
generic fault 58
Input transformation 47
logs 88
manually creating 38
map ports to anyTypes 77
monitoring 88
operation properties 83
operations 13
overview 12
overview view 34
permission types 85
permissions 85
predefined faults 57
process 12
properties 82
property configuration 81
security 84
substitution groups 78
system-defined fault 57
trace level 88
WSDL 14
WSDL association 37
WSDL URL 14
WSDL view 36
Web Service Consumer transformation
adding HTTP headers 95
advanced properties 102
concurrent web service request messages 102
cookie authentication 95
creating 108
dynamic web service URL 95
dynamic WS-Security name 95
early selection optimization 106
enabling generic fault output 102
enabling HTTP error output 102
enabling push-into optimization 107
endpoint URL 95
error handling 104
filter optimization 106
generic SOAP faults 104
input mapping 96
Web Service Consumer transformation *(continued)*
mapping input ports 96
mapping output nodes 99
operations 92
output mapping 99
overview 91
push-into optimization 106
security 93
SOAP compression 104
SOAP messages 92
transport layer security 93
viewing keys 97, 100
web service security
authentication 84
authorization 84
HTTP client filter 84
HTTPS 84
message layer security 84
pass-through security 84
permissions 84
transport layer security 84
web service transformations
Location column 70
web services connections
overview 102
WS-Security user name
dynamic port 95
WSDL
association with web service 37
create a web service from a WSDL 37
description 14
WSDL data objects
advanced view 18
creating 18
importing 17
overview view 18
schema view 17
synchronization 19
WSDL file
binding element 92
operation element 92
port element 92
service element 92
WSDL URL
description 14