

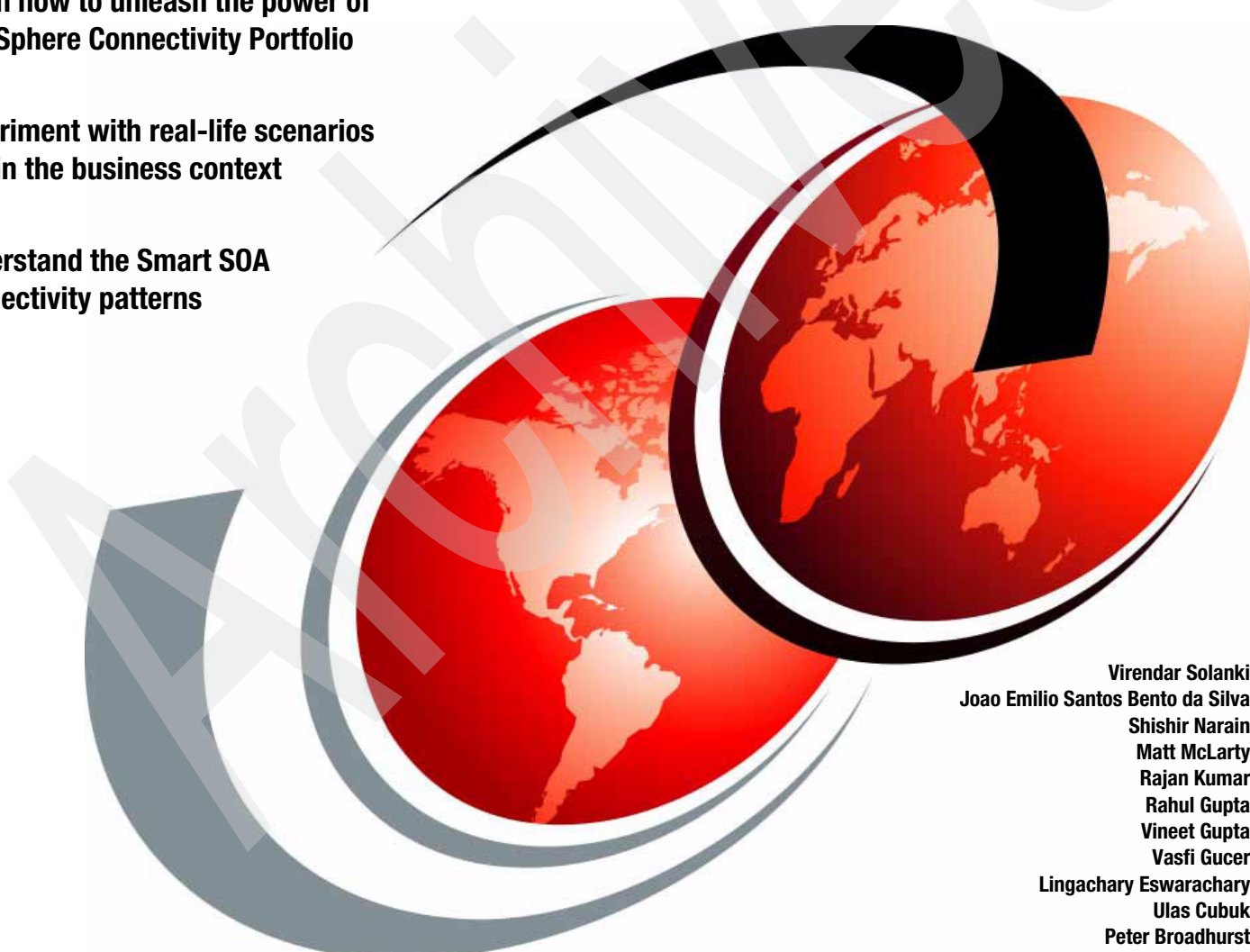
Smart SOA Connectivity Patterns

Unleash the Power of IBM WebSphere Connectivity Portfolio

Learn how to unleash the power of WebSphere Connectivity Portfolio

Experiment with real-life scenarios within the business context

Understand the Smart SOA connectivity patterns



Virendar Solanki
Joao Emilio Santos Bento da Silva
Shishir Narain
Matt McLarty
Rajan Kumar
Rahul Gupta
Vineet Gupta
Vasfi Gucer
Lingachary Eswarachary
Ulas Cubuk
Peter Broadhurst



International Technical Support Organization

**Smart SOA Connectivity Patterns: Unleash the power
of IBM WebSphere Connectivity Portfolio**

September 2011

Archived

Note: Before using this information and the product it supports, read the information in “Notices” on page xxv.

First Edition (September 2011)

This edition applies to:

IBM WebSphere MQ Version 7.0.1.5
IBM WebSphere Integration Designer Version 7.5
IBM WebSphere Enterprise Service Bus Version 7.5
IBM WebSphere Message Broker Version 7.0.0.2
IBM WebSphere Services Registry and Repository Version 7.5
IBM WebSphere MQ File Transfer Edition Version 7.0.4.0
IBM WebSphere DataPower Integration Appliance XI50

© Copyright International Business Machines Corporation 2011. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contact an IBM Software Services Sales Specialist



Start SMALL, Start BIG, ... **JUST START** architectural knowledge, skills, research and development . . . **that's IBM Software Services for WebSphere.**

Our highly skilled consultants make it easy for you to design, build, test and deploy solutions, helping you build a smarter and more efficient business. **Our worldwide network of services specialists wants you to have it all!** Implementation, migration, architecture and design services: IBM Software Services has the right fit for you. We also deliver just-in-time, customized workshops and education tailored for your business needs. You have the knowledge, now reach out to the experts who can help you extend and realize the value.

For a WebSphere services solution that fits your needs, contact an IBM Software Services Sales Specialist:
ibm.com/developerworks/websphere/services/contacts.html

Archived

Contents

Contact an IBM Software Services Sales Specialist	iii
Figures	ix
Tables	xix
Examples	xxi
Notices	xxv
Trademarks	xxvi
Preface	xxvii
The team who wrote this book	xxvii
Now you can become a published author, too!	xxxii
Comments welcome	xxxii
Stay connected to IBM Redbooks	xxxii
Chapter 1. Introduction	1
1.1 Scope	2
1.1.1 Business versus technology	2
1.1.2 Products in scope	2
1.2 What is Smart SOA	8
1.2.1 Business value of the Smart SOA approach	8
1.2.2 IT value of the Smart SOA approach	9
1.2.3 The right Smart SOA style for you	9
1.2.4 Learn more	9
1.3 How to use this book	10
1.3.1 ITSO Enterprise	10
1.3.2 Contents	11
1.3.3 Connectivity patterns that are used in the book chapters	22
1.4 Intended audience	22
Chapter 2. Service Enablement pattern	23
2.1 Introduction to the business scenario	24
2.1.1 Scenario problem	24
2.1.2 Solution	25
2.2 Architectural diagram	26
2.3 Benefits	26
2.3.1 Enterprise Service Bus features using WebSphere ESB	27
2.4 Technical implementation	27
2.4.1 Importing the ITSO library into the IBM Integration Designer workspace	27
2.4.2 Building the mediation module for Add/Update/Find customers	30
2.4.3 Assembling the mediation module	33
2.4.4 Creating a JDBC outbound adapter	36
2.4.5 Implementing the mediation module	53
2.4.6 Testing the mediation module	67
2.5 Summary	73
Chapter 3. Service Enablement pattern: Enterprise Content Management System ..	75
3.1 Introduction to the business scenario	76

3.1.1 Scenario problem	76
3.1.2 Scenario solution	76
3.2 Assembly Diagram	76
3.3 Benefits	77
3.4 Technical implementation	77
3.4.1 Configuring FileNet with custom Document Class type to manage legal contract documents electronically	78
3.4.2 Using WebSphere Adapter for Enterprise Content Management to service-enable FileNet content	88
3.4.3 Building an FTP interface for the createDocument business service exposed via WebSphere Adapter	112
3.4.4 Building a web application on the retrieveAllDocuments service to query FileNet documents	135
3.4.5 Adding other protocols or application-based interfaces for other services	147
3.5 Summary	148
Chapter 4. Service Reuse and Governance pattern	149
4.1 Introduction to the business scenario	151
4.1.1 Scenario problem	151
4.1.2 Scenario solution	152
4.2 Architectural diagram	153
4.3 Benefits	153
4.3.1 Technical implementation	154
4.3.2 Enabling governance in WebSphere Service Registry and Repository for contract enforcement	155
4.3.3 IBM WebSphere ESB configuration to establish connection with WebSphere Service Registry and Repository	175
4.3.4 Developing the mediation module	181
4.4 Summary	198
Chapter 5. Message Privacy Enforcement pattern	201
5.1 Introduction to the business scenario	203
5.1.1 Business requirements	203
5.1.2 Available options	204
5.2 Architecture diagram	204
5.2.1 Pattern	205
5.2.2 Products used	206
5.2.3 Key features	206
5.2.4 Solution diagram	206
5.3 Benefits	207
5.4 Technical implementation	208
5.4.1 Assumptions	208
5.4.2 Prerequisites	208
5.4.3 Setting up the environment	209
5.4.4 Preparing to create storeusr key database and certificates	212
5.4.5 Creating a wesbusr key database and certificates	216
5.4.6 Exchanging certificates	220
5.4.7 Implementing the solution	223
5.5 Summary	232
Chapter 6. Enterprise Messaging and File Backbone pattern	233
6.1 Introduction to the business scenario	234
6.1.1 Limitations with file transfer protocol	234
6.1.2 Problem scenario	235

6.1.3	Managed File Transfer offering from IBM	238
6.2	Architectural diagram	240
6.2.1	Enterprise Messaging and File Backbone pattern	241
6.2.2	Products utilized to implement this pattern	242
6.2.3	Addressing ITSO Enterprise issues with WebSphere MQ File Transfer Edition	242
6.2.4	Systems utilized to implement this scenario	243
6.3	Benefits	244
6.3.1	Short-term benefits	244
6.3.2	Long-term benefits	245
6.4	Technical implementation	245
6.4.1	Topology	246
6.4.2	Preparing WebSphere MQ	247
6.4.3	Preparing WebSphere MQ File Transfer Edition	255
6.5	Business scenario 1	266
6.5.1	Requirement	266
6.5.2	Solution	266
6.5.3	Testing checkpoint and restart	267
6.6	Business scenario 2	268
6.6.1	Requirement	269
6.6.2	Solution	269
6.6.3	Testing task automation	283
6.7	Business scenario 3	287
6.7.1	Requirement	288
6.7.2	Solution	288
6.7.3	Testing the file transfer audit feature	291
6.8	Business scenario 4	296
6.8.1	Requirement	296
6.8.2	Solution	297
6.8.3	Testing centralized administration	298
6.9	Business scenario 5	301
6.9.1	Requirement	301
6.9.2	Solution	301
6.9.3	Testing the SSL configuration	314
6.10	Summary	315
	Chapter 7. Evolve to SOA pattern	317
7.1	Introduction to the business scenario	318
7.1.1	Current implementation	319
7.1.2	Technical challenges	322
7.1.3	Evolving to service-oriented architecture (SOA)	322
7.2	SOA solution	323
7.2.1	Products utilized to implement this pattern	325
7.3	Benefits	325
7.4	Technical implementation	326
7.4.1	Database setup	326
7.4.2	MQ setup	327
7.4.3	Broker setup	327
7.4.4	Message flow development with the Message Broker Toolkit	328
7.4.5	Testing the message flows	350
7.5	Summary	351
	Chapter 8. Hybrid Bus pattern	353
8.1	Introduction to the business scenario	354

8.2 Architectural diagram	354
8.2.1 Scenario 1: Expose Order Web Service	354
8.2.2 Scenario 2: Secure order fulfillment and order status messages	355
8.3 Benefits	355
8.4 Technical implementation	355
8.4.1 Ordering Web Service Proxy	355
8.4.2 Payment Web Service Proxy	379
8.4.3 Creating a multi-protocol gateway	385
8.5 Summary	394
Chapter 9. Conclusion	395
9.1 Benefits of Smart SOA	396
9.2 Business achievements	396
9.2.1 Achievement from scenario 1	396
9.2.2 Achievement from scenario 2	396
9.2.3 Achievement from scenario 3	397
9.2.4 Achievement from scenario 4	397
9.2.5 Achievement from scenario 5	397
9.2.6 Achievement from scenario 6	397
9.2.7 Achievement from scenario 7	397
9.3 What is next for ITSO Enterprise in the Smart SOA ladder	398
9.4 Resources	398
Appendix A. Additional material	401
Locating the web material	401
Using the web material	401
System requirements for downloading the web material	401
Downloading and extracting the web material	402
Related publications	403
IBM Redbooks publications	403
Online resources	403
Help from IBM	404

Figures

1-1 Business value of the Smart SOA approach	9
1-2 IT value of the Smart SOA approach	9
1-3 ITSO Enterprise customer channels and stale data	11
1-4 Service Enablement	12
1-5 Service Enablement	12
1-6 Business context for Service Enablement of ECM	13
1-7 ITSO Enterprise: CSR providing order information	14
1-8 Service Governance with SLA	15
1-9 Authorization Service Messaging System	16
1-10 Message encryption	16
1-11 Enforce message privacy	17
1-12 Unmanaged file transfer currently deployed in the ITSO Enterprise company	18
1-13 Managed File Transfer features	18
1-14 Enterprise Messaging and File Backbone pattern	19
1-15 Evolve to SOA	20
1-16 Hybrid Bus pattern	21
1-17 Smart SOA solution	21
2-1 ITSO Enterprise infrastructure	24
2-2 IBM WebSphere ESB solution for ITSO Enterprise infrastructure	25
2-3 An architectural diagram with a WebSphere ESB solution for ITSO Enterprise	26
2-4 Import dialog box	28
2-5 Import Project Interchange Contents dialog	29
2-6 Library project view	29
2-7 Create the mediation module	30
2-8 Mediation module name and target runtime environment	31
2-9 Library selection	31
2-10 Business object parsing mode	32
2-11 Assembly Diagram	32
2-12 Drag the interface to the Assembly Diagram	33
2-13 Component Creation	33
2-14 Select a Transport Protocol	34
2-15 Web service binding	35
2-16 Web Service Ports in library and Web service symbol in the Assembly Diagram	36
2-17 Assembly Diagram	36
2-18 Run admin console	37
2-19 Log in to the console	38
2-20 Global security	38
2-21 J2C authentication data	39
2-22 New authentication	39
2-23 Authentication entry	40
2-24 Save the configuration	40
2-25 JDBC providers	41
2-26 Select DB2 Universal JDBC Driver Provider (XA)	41
2-27 Select the data source	42
2-28 Create new data source	42
2-29 Enter basic data source information	43
2-30 Database specific properties for the data source	43
2-31 Security alias	44

2-32	Review all properties	44
2-33	Save changes to the master configuration	45
2-34	Test the connection for the data source	45
2-35	Success message	46
2-36	Three data source configurations for three separate databases	46
2-37	JDBC outbound adapter	47
2-38	Select JDBC adapter	47
2-39	Select the target runtime environment	48
2-40	Locate JAR files	48
2-41	Database connection properties	49
2-42	Find the objects.	50
2-43	Select database table	50
2-44	Select operations to add to the service interface	51
2-45	Specify predefined XA data source.	51
2-46	Specify the Location Properties	52
2-47	Assembly Diagram	52
2-48	Assembly Diagram with end-to-end connectivity	53
2-49	Operation connections window	53
2-50	Add customer	54
2-51	Service Integration	54
2-52	Select the services	54
2-53	Mediation flow	55
2-54	Wired mediation primitives in flow.	56
2-55	Mediation flow	56
2-56	Create an XML Map	57
2-57	Specify Message Types	57
2-58	XSLT map.	57
2-59	XSLT mapping source to target	58
2-60	Mediation flow	59
2-61	New Business Object Map	59
2-62	Specify Message Types	60
2-63	BO map.	60
2-64	Submap in New Business Object Map	60
2-65	BO Map with Custom transform	61
2-66	Java code in mapping to print response message	61
2-67	Mediation flow for add customers	61
2-68	Interface representation	62
2-69	Select the services	62
2-70	Assign transform properties	63
2-71	Mediation flow for update customer details.	63
2-72	Interface representation	64
2-73	Select a Mediation Flow Template	64
2-74	CustomerInfoService interface	64
2-75	CustomerInfoService interface properties.	65
2-76	Mediation flow: Find customer details.	65
2-77	Database lookup properties	66
2-78	Specify Message Types	66
2-79	XSLT mapping	67
2-80	Mediation Flow: Find Customer details.	67
2-81	Start WebSphere ESB Server v7.5.	67
2-82	Add and Remove mediation module.	68
2-83	Add and Remove	69
2-84	After the deployment on the ESB Server	69

2-85 Console	70
3-1 Higher-level block diagram of the solution involving WebSphere Adapter for Enterprise Content Management	77
3-2 FileNet business services exposed via WebSphere Adapter for Enterprise Content Management	78
3-3 Create new document class	79
3-4 Enter a valid name for the Document Class: Supplier Contract	79
3-5 Select Document as the superclass for the document class type that we will create	80
3-6 Page to configure document properties	81
3-7 Enter the name for the property	81
3-8 Select the data type for the property	82
3-9 Optionally assign a choice list for the property	82
3-10 Choose the type of property as either single or multi	83
3-11 Summary page for property creation	83
3-12 Final list of properties added to the document class	84
3-13 Inheritance settings to be configured from the chosen parent class	85
3-14 Auditing definitions for the document class to be created	86
3-15 Summary page for class creation wizard	87
3-16 Supplier Contract document class successfully created on the repository AKOS	87
3-17 Create a new mediation module to store project artifacts	88
3-18 Enter a name for the mediation module	89
3-19 Choose Lazy parsing for the parsing mode for the mediation module	89
3-20 Drag and drop the ECM control onto the Assembly Diagram	90
3-21 Select the adapter for the FileNet discovery process	91
3-22 Importing the adapter's binary RAR file into the workspace	92
3-23 Select the runtime dependency file from the local WebSphere ESB installation directory	93
3-24 Enter the FileNet connection information	94
3-25 Adapter discovery process connecting to the FileNet system	95
3-26 Adapter displays the FileNet content in a metadata tree structure	96
3-27 Customize the document class properties for the selected object	97
3-28 Select document class is added successfully to the Selected objects list panel	98
3-29 Target relative folder name to store generated business objects	99
3-30 Configure service generation and deployment properties	100
3-31 Choose the target server to configure the JAAS alias	101
3-32 Choose the JAAS alias to connect to FileNet	101
3-33 External service generating required artifacts	102
3-34 Name the service FileNetServices	103
3-35 Add the project to the server for deployment	104
3-36 Prepare the project for deployment	105
3-37 Click Finish to complete the project deployment process	106
3-38 Project successfully deployed to server	106
3-39 Launch the Integration Test Client component of WebSphere ESB	107
3-40 Integration Test Client to supply values for validation	108
3-41 Integration Test Client populated with test values	109
3-42 Invoke the validation process	109
3-43 Choose the deployment location for testing the component	110
3-44 Integration Test Client running test	110
3-45 The response from the service exposed via adapter is shown on the editor	111
3-46 Document successfully created on FileNet	112
3-47 Create a new Business Object	113
3-48 Name the business object SupplierContract	113
3-49 Add fields to the business object	114

3-50	Final structure of business object	114
3-51	Initiate the FTP adapter's discovery process	115
3-52	Choose predefined template approach for faster integration with FTP server	116
3-53	Enter the service Name FTPExport	117
3-54	Populate the required fields for the FTP service	118
3-55	Configure the security credential for connecting to the FTP server	119
3-56	Choose the input file format	120
3-57	Choose the Delimited data format of the input file	121
3-58	Type the Delimiter for the input file processing	122
3-59	Data handler configuration	123
3-60	Complete the data handler configuration	124
3-61	Service for FTP successfully generated and added to the Assembly Diagram	124
3-62	Add a Mediation Component to the project	125
3-63	Wire the FTP service and the Mediation component	125
3-64	Wire the Mediation component and the FileNet service	126
3-65	Start implementing the Mediation component	126
3-66	Choose a folder for the Mediation component's implementation	127
3-67	Integrate the FTP and FileNet services	127
3-68	Select createAKOSSupplierContract for the target operation	128
3-69	Select the service for integration	128
3-70	XSLTransformation1 component for mapping two separate formats	129
3-71	Specifying that the XSLTransformation1 component will use the target schema's default/fixed values	129
3-72	Map the elements of the source service and the target service	130
3-73	Assign a default value for the RepositoryId field	131
3-74	Assign the value for an enum field	131
3-75	Assign a valid value for enum-type fields	132
3-76	Concat function for constructing the cmisU58name field	133
3-77	Wire the response back to the FTP service	133
3-78	Sample values to validate the flow	134
3-79	Result of the final execution	134
3-80	Create a new Interface	135
3-81	Enter the Name for the interface	136
3-82	Interface with one operation queryDocument	136
3-83	Drag and drop the WSEExport interface on the Assembly Diagram to start building the Web service binding	137
3-84	Choose the transport protocol for the Web service binding	137
3-85	Provide the target namespace for the Web service binding	138
3-86	Wire the Web service component to the existing Mediation component	138
3-87	Assembly Diagram of all services	139
3-88	Add the newly created interface WSEExport to the mediation module	139
3-89	Choose Service Integration as the mediation flow	140
3-90	Choose a Reference and a Target Operation to integrate the services	140
3-91	Wire the services using the XSLTransformation1 component	141
3-92	Map the necessary fields	141
3-93	Connect the response flow of the Mediation component	142
3-94	Auto mapping of input fields to output fields	142
3-95	Configuration for advanced auto mapping functionality	143
3-96	Final structure of the map with all the fields mapped	143
3-97	Launch the Web Service Client generation wizard	144
3-98	Web Service Client generation configuration	145
3-99	Additional settings for Web Service Client generation	146
3-100	Generation of artifacts and code required for the client and updating them on the	

server	147
4-1 ITSO Enterprise current query order service	151
4-2 WebSphere ESB and WebSphere Service Registry and Repository Solution for ITSO Enterprise	152
4-3 Assembly Diagram of WebSphere ESB and WebSphere Service Registry and Repository solution	153
4-4 Loading documents	156
4-5 Loading the WSDL file	157
4-6 Dependent XSD documents of ITSOOrdering.wsdl	157
4-7 Capability life cycle diagram	158
4-8 Creating New Business Service	158
4-9 Business Service details	159
4-10 Edit Business Service Relationships	159
4-11 Adding Charter document to Orders Service	160
4-12 Search for an existing organization as a target owning organization	161
4-13 Business Capability Approved governance state	161
4-14 SOA life cycle	162
4-15 Viewing Approved Business Capabilities	162
4-16 Adding Service Version	163
4-17 Service version details	163
4-18 Updating the Provided Web Services relationship	164
4-19 Governance Policy Validator	165
4-20 Proposing Realization	165
4-21 Approving Realization	166
4-22 Service Versions	166
4-23 Adding a New SLD to a service version	166
4-24 Editing the SLD properties	167
4-25 Adding additional QoS information to the SLD	168
4-26 Adding service endpoint that realizes the SLD	169
4-27 Finalizing the SLD relationships update	169
4-28 Viewing the life cycle history	170
4-29 Edit Classifications	171
4-30 Adding the Staging classification to the service endpoint	171
4-31 Approving the endpoint for use	171
4-32 Proposing Staging Deployment	171
4-33 Assigning Consumer Identifier to consumer service version	173
4-34 Adding the target SLA of Consumes relationship of the service version	174
4-35 Agreed Endpoints relationship	174
4-36 SLA Governance State	175
4-37 WSRR definitions	176
4-38 WSRR definition Configuration page	176
4-39 WSRR definition connection properties	177
4-40 Create a new authentication alias	178
4-41 SSL configuration	178
4-42 Testing connection to WSRR instance	179
4-43 Service Registries category	180
4-44 WebSphere Service Registry and Repository Connection Configuration	180
4-45 Import Projects interchange dialog	182
4-46 Library project view	183
4-47 Library selection	183
4-48 Drag the interface to the Assembly Diagram	184
4-49 Component Creation	184
4-50 Assembly Diagram	185

4-51	Drag reference interface to the Assembly Diagram	185
4-52	Component Creation	186
4-53	Select the Web service port	186
4-54	Assembly Diagram	186
4-55	Assembly Diagram with wired components	187
4-56	Operation connections window	187
4-57	Query order	187
4-58	Service integration	188
4-59	Select the services	188
4-60	Request mediation flow	189
4-61	Wired mediation primitives in request flow	189
4-62	MessageLog properties	190
4-63	SLAEndpointLookup properties	190
4-64	Create an XML Map	190
4-65	Specify Message Types	191
4-66	XSLT mapping for CSRQueryOrderTOOrderQuery	191
4-67	New XML Map	192
4-68	XSLT map for NoMatch	192
4-69	Request flow	193
4-70	Response flow	193
4-71	Trace primitive properties	194
4-72	Create an XML Map	194
4-73	Specify Message Types	194
4-74	XSLT map for ResponseMap	195
4-75	Response flow	195
4-76	Start WebSphere ESB Server v7.5	195
4-77	Add and Remove mediation module	196
4-78	Add and Remove	196
4-79	After deployment on ESB Server	197
4-80	Console	197
5-1	ITSO Enterprise hub and spoke design: Stores and back-end system	203
5-2	Architectural design	204
5-3	Message Privacy Enforcement	205
5-4	Security gateway	206
5-5	Solution diagram	207
5-6	Run as different user	212
5-7	storeusr login	212
5-8	IBM Key Management	213
5-9	New CMS key database for storeusr	214
5-10	Password prompt for the storeusr.kdb file	214
5-11	Stash file information message	215
5-12	New Self-Signed Certificate for the storeusr	215
5-13	CMS key database personal certificate list updated	216
5-14	Run Command Prompt as different user	216
5-15	Log in as wesbusr	217
5-16	IBM Key Management	218
5-17	Create a new JKS key database	218
5-18	Key database password set	219
5-19	JKS key database Personal Certificates list	219
5-20	Create New Self-Signed Certificate window	220
5-21	JKS key database personal certificates list updated	220
5-22	Import certificate from storeusr.kdb	221
5-23	Password dialog for importing the CMS certificate	221

5-24	storeusr.kdb certificate list	221
5-25	Import key Change Labels window	222
5-26	webusr.jks Personal Certificates list updated	222
6-1	ITSO Enterprise warehouse and stores	236
6-2	ITSO Enterprise traditional file transfer implementation	237
6-3	Managed File Transfer features	238
6-4	Overview of WebSphere MQ File Transfer Edition	239
6-5	Managed File Transfer using WebSphere MQ File Transfer Edition	240
6-6	Enterprise Messaging and File Backbone pattern for ITSO Enterprise	241
6-7	Servers that were used by ITSO Enterprise at various locations	244
6-8	Integrating Managed File Transfer as part of WebSphere MQ network	245
6-9	WebSphere MQ topology for File Transfer Edition	246
6-10	Cluster channels in MQ cluster INVENTORY.CLUSTER	252
6-11	Cluster administration from WebSphere MQ Explorer	254
6-12	wmqfte.properties file	257
6-13	coordination.properties file	257
6-14	command.properties file	258
6-15	agent.properties for Atlanta store File Transfer Edition agent	260
6-16	agent.properties file for Chicago store File Transfer Edition agent	261
6-17	agent.properties for NEWYORK store File Transfer Edition agent	261
6-18	agent.properties file for warehouse File Transfer Edition agent	262
6-19	fteListAgents output for ITSO Enterprise file agents	263
6-20	Agent details in WebSphere MQ Explorer	264
6-21	Transfer log for file transfer validation	266
6-22	The current transfer state	267
6-23	Disable the network on the warehouse	268
6-24	The current transfer state after the network is enabled	268
6-25	Test results for the transfer.xml ant script	272
6-26	Updated agent.properties for File Transfer Edition agent STORE.ATLANTA.QM	279
6-27	Inventory and trigger files for the ITSO Enterprise Atlanta store	283
6-28	Inventory and trigger files for the ITSO Enterprise Chicago store	283
6-29	Inventory and trigger files for the NEWYORK store	284
6-30	Transfer log for the files that are transferred from the stores to the warehouse	284
6-31	Daily inventory file that was created in the warehouse	284
6-32	Daily inventory file that was copied for the Inventory processing system	285
6-33	Daily inventory file moved into ARCHIVE repository	285
6-34	Transfer log for the task automation failure use case	287
6-35	Email sent to Atlanta store admin stating file transfer failures	287
6-36	Database Logger installation window	289
6-37	XA configuration on ITSO.FILE.COORDINATION.QM	289
6-38	The databaselogger.properties file for the File Transfer Edition database logger	290
6-39	File transfer events archived in database FTELOG	292
6-40	MQ Explorer plug-in for the File Transfer Edition	297
6-41	File Transfer Edition agent details in WebSphere MQ Explorer	298
6-42	ITSO Enterprise warehouse agent status is Stopped	299
6-43	Agent connectivity test from MQ Explorer	299
6-44	Connectivity test for ITSO Enterprise Store agent STORE.ATLANTA.QM	300
6-45	File transfer progress	300
6-46	The agent.properties file for the ITSO Atlanta store File Transfer Edition agent	311
6-47	The agent.properties file for the ITSO Chicago store File Transfer Edition agent	311
6-48	Agent.properties for NEWYORK store File Transfer Edition agent	312
6-49	The agent.properties file for the ITSO Enterprise warehouse agent	312
6-50	command.properties	313

6-51	coordination.properties	313
6-52	File transfer results with SSL configuration.	314
7-1	Overview of the current supplier integration	319
7-2	Supplier integration for ITSO Enterprise	325
7-3	Creating a new message set for csv to XML transformation.	328
7-4	Selecting the type of message data	329
7-5	Schema definitions file for csv to XML conversion	330
7-6	Creating new message flow	331
7-7	Pattern parameters	332
7-8	Message flow for ITSO_Warehouse_CreateOrder_MessageFlow.	333
7-9	Properties for InvOutput Supplier Orders node.	333
7-10	Subflow ITSO_Warehouse_OrderProcessor_subflow	334
7-11	Subflow ITSO_Warehouse_Router_subflow	335
7-12	Web Service URL for the HTTP Request Node	337
7-13	OrderQ MQ Output Node properties.	338
7-14	Create a new message set for the Order Status flow	340
7-15	Message data options for the message set	341
7-16	Importing the WSDL for the message definition	342
7-17	Message set project after the WSDL import	343
7-18	Creating a new message flow for the Order Status project.	344
7-19	Message flow ITSO_Warehouse_OrderStatus_MessageFlow.	344
7-20	Properties configuration for OrderStatusInput node	345
7-21	Message subflow OrderStatusUpdate_SubFlow	345
7-22	Configuration for the Extract node	346
7-23	Creating the message map.	347
7-24	Specifying the source and target for the message map	348
7-25	Mapping editor for the message map	349
7-26	Configuration for UpdateDB Data Insert node	349
7-27	ITSO.JMS.90045.QUEUE.	350
7-28	ITSO3 EDI order message	351
8-1	WebSphere DataPower	354
8-2	WebSphere DataPower	355
8-3	End-to-end processing	356
8-4	Web Service Proxy	357
8-5	Creating a new Web Service Proxy	357
8-6	Adding WSDL to the Web Service Proxy	358
8-7	Uploading the WSDL file.	358
8-8	Adding local endpoint handler.	359
8-9	Configuring HTTPS (SSL) Front Side Handler	359
8-10	Configuring the advanced filtering parameters	360
8-11	Assigning the SSL Proxy	360
8-12	Adding the local endpoint handler.	361
8-13	WS-Proxy status	361
8-14	Generating a private key and self-signed certificate using Crypto Tools	362
8-15	Keys in the file system	363
8-16	Keys and Certificate Management	363
8-17	Keys and Certificate Management Page	364
8-18	Configure Crypto Identification Credentials	364
8-19	Configure Crypto Validation Credentials.	365
8-20	Configure Crypto Profile	366
8-21	Configure SSL Proxy Profile	367
8-22	AAA framework	368
8-23	WSDL Policy Tree Representation	369

8-24	AAA action	369
8-25	Configure an AAA Action	370
8-26	Define how to extract a user's identity from an incoming request.	370
8-27	Define how to authenticate the user	371
8-28	Choose post-processing	372
8-29	Content-based routing	373
8-30	Configure Web Service Proxy settings	374
8-31	Defining the dynamic back-end behavior of WS-Proxy.	374
8-32	Adding the Route action	375
8-33	Configuring the routing selection method	375
8-34	Assigning high priority	377
8-35	Opening the MQ Queue Manager list page	378
8-36	Configure MQ Queue Manager.	379
8-37	End-to-end processing for Payment Proxy	380
8-38	Opening Processing Rules	381
8-39	Drag and drop the Encrypt action	381
8-40	Configuring the Encrypt Action Message Type as field level	382
8-41	Opening XPath Tool for guidance to build the XPath expression	382
8-42	Building the XPath expression	383
8-43	Selecting the recipient certificate	384
8-44	SLM Policy tab	385
8-45	SLM Request rule	385
8-46	Control Panel	386
8-47	General Configuration section	386
8-48	Configure MQ Front Side Handler window	387
8-49	Configure MQ Queue Manager.	388
8-50	Configure HTTP Front Side Handler.	389
8-51	Front side settings window	389
8-52	Create SupplierWSRule	390
8-53	Create the processing actions	390
8-54	Create MQRule	391
8-55	Create the processing actions	391
8-56	Create OrdersTestMPG_Policy1_rule_0	392
8-57	Three rules shown	392
8-58	Select PaymentClientCryptoProfile.	393
8-59	Configure Multi-Protocol Gateway window	393
8-60	Configure Multi-Protocol Gateway window	394

Archived

Tables

1-1	Connectivity patterns that are used in the book chapters	22
3-1	Required properties for the document class	84
3-2	Mandatory fields	108
5-1	setmqspl flags	227
6-1	Components for Enterprise Messaging and File Backbone pattern	246
6-2	Participating queue manager in cluster INVENTORY.CLUSTER	251
6-3	Update agent.properties files with commandPath	279
6-4	File Transfer Edition agent status values	298
7-1	ITSO suppliers	320
9-1	IBM resources for Smart SOA	399

Archived

Examples

2-1 SOAP request message: Add customer details	70
2-2 SOAP Response message: Add customer details	71
2-3 SOAP Request Message: Update Customer details	71
2-4 SOAP Response Message: Update Customer details	72
2-5 SOAP Request message: Find Customer details	72
2-6 SOAP Response Message: Find Customer details	72
3-1 Sample supplier contract file content	134
4-1 SOAP request message to query an order for a specific customerID.	197
4-2 SOAP response message: Order details for a specific customerID	198
5-1 Create AMS.QM queue manager output	209
5-2 Starting the queue manager output	209
5-3 Creating a listener output	209
5-4 Creating a channel output.	210
5-5 Creating queues output.	210
5-6 Creating MQ AMS system queues	210
5-7 Granting access to users to connect to the queue manager	211
5-8 Granting access to the store user to send messages	211
5-9 Granting access to the ESB to read messages	211
5-10 Granting browse access on the AMS system queues	211
5-11 Command prompt as storeusr output	212
5-12 strmqikm	213
5-13 Command prompt as wesbusr output.	217
5-14 strmqikm	217
5-15 Sending a test message to the queue	223
5-16 Getting the test message from the queue.	223
5-17 Sending a message as Administrator	223
5-18 Getting a message as Administrator.	224
5-19 MQPut.java	224
5-20 MQGet.java	225
5-21 MQPut output	226
5-22 MQGet program output	226
5-23 Creating the protection policy	226
5-24 Enabling server interceptor.	227
5-25 Enabling the Java interceptor	227
5-26 AMS protection policy in action.	228
5-27 CMS Key DB config file.	229
5-28 JKS Key DB configuration file.	229
5-29 MQPut.bat	230
5-30 MQGet.bat	230
5-31 MQput.bat output	230
5-32 MQGet.bat output	231
6-1 The crtmqm command syntax.	247
6-2 Create queue manager STORE.ATLANTA.QM	247
6-3 Create queue manager STORE.CHICAGO.QM	247
6-4 Create queue manager STORE.NEWYORK.QM	248
6-5 Create queue manager WAREHOUSE.AGENT.QM	248
6-6 Create queue manager ITSO.FILE.COORDINATION.QM	248
6-7 Create queue manager ITSO.FILE.COMMAND.QM	248

6-8	The syntax of the strmqm command	249
6-9	Start queue manager commands	249
6-10	Create listener on the queue manager STORE.ATLANTA.QM	249
6-11	Create listener on the queue manager STORE.CHICAGO.QM	249
6-12	Create listener on the queue manager STORE.NEWYORK.QM	250
6-13	Create listener on the queue manager WAREHOUSE.AGENT.QM	250
6-14	Create listener on the queue manager ITSO.FILE.COORDINATION.QM	250
6-15	Create listener on the queue manager ITSO.FILE.COMMAND.QM	250
6-16	Start queue manager listener objects	250
6-17	Change queue manager to full repository	252
6-18	Cluster channels on ITSO.FILE.COMMAND.QM	252
6-19	Cluster channels on ITSO.FILE.COORDINATION.QM	253
6-20	Cluster channels on STORE.ATLANTA.QM	253
6-21	Cluster channels on STORE.CHICAGO.QM	253
6-22	Cluster channels on STORE.NEWYORK.QM	253
6-23	Cluster channels on WAREHOUSE.AGENT.QM	254
6-24	Server connection channel	255
6-25	The syntax for the fteSetupCoordination command	256
6-26	Set up the coordination properties	256
6-27	Configuring the coordination queue manager manually	257
6-28	Configuring the coordination queue manager using an mqsc script	257
6-29	The syntax for the fteSetupCommands command	258
6-30	fteSetupCommands command properties	258
6-31	The syntax for fteCreateAgent command	259
6-32	fteCreateAgent command on the Atlanta store	259
6-33	Create MQ objects for Atlanta agent	260
6-34	fteCreateAgent on the Chicago store	260
6-35	Create MQ objects for the Chicago agent	260
6-36	fteCreateAgent on the NEWYORK store	261
6-37	create MQ objects for the NEWYORK agent	261
6-38	fteCreateAgent on the warehouse	262
6-39	Create MQ objects	262
6-40	The syntax for the fteStartAgent command	262
6-41	Start agents on the stores and the warehouse	263
6-42	The syntax for fteListAgents command	263
6-43	The syntax for ftePingAgent command	264
6-44	The syntax for command fteCreateTransfer	264
6-45	Inventory file transfer from the Atlanta store to the warehouse	265
6-46	Inventory file transfer from the Chicago store to the warehouse	265
6-47	Inventory file transfer from the NEWYORK store to the warehouse	265
6-48	File transfer request from the Chicago store to the warehouse	267
6-49	The syntax for the fteAnt command	269
6-50	Ant script transfer.xml for the ITSO Enterprise Atlanta store	270
6-51	Using the fteant command to test the transfer.xml Ant script for the Atlanta store	271
6-52	Ant script transfer.xml for the Chicago store	272
6-53	Ant script Transfer.xml for the NEWYORK store	273
6-54	Ant script concatenate.xml for the warehouse	275
6-55	The syntax for fteCreateMonitor command	276
6-56	Creating inventoryTransferTask for the ITSO Enterprise Atlanta store	277
6-57	Creating inventoryTransferTask for the ITSO Enterprise Chicago store	277
6-58	Creating inventoryTransferTask for the ITSO Enterprise Newyork store	278
6-59	Creating inventoryConcatenateTask.xml for the ITSO Enterprise warehouse	278
6-60	Monitor for ITSO Enterprise Atlanta store	279

6-61	Monitor for ITSO Enterprise Chicago store	280
6-62	Monitor for ITSO Enterprise NEWYORK store	280
6-63	Monitor for the ITSO Enterprise warehouse	280
6-64	The syntax for the command fteListMonitors	281
6-65	List of resource monitors that have been created on the ITSO stores and the warehouse	281
6-66	Scheduled inventory file transfer to the Inventory processing system	282
6-67	Scheduled inventory file transfer to the archive repository	282
6-68	Scheduled transfer of the ITSO Enterprise inventory file	282
6-69	New transfer.xml created for Atlanta store to create a failure	285
6-70	The syntax for the fteStartDatabaseLogger command	291
6-71	The fteStartDatabaseLogger command	291
6-72	FTEAudit.java for querying the database FTELOG	293
6-73	Syntax to run FTEAudit.java	294
6-74	Output from the execution of program FTEAudit.java	294
6-75	Stop the File Transfer Edition agent	299
6-76	Create truststore and keystore for Atlanta File Transfer Edition agent	302
6-77	Create truststore and keystore for Chicago File Transfer Edition agent	302
6-78	Create truststore and keystore for the New York File Transfer Edition agent	302
6-79	Create truststore and keystore for warehouse File Transfer Edition agent	302
6-80	Command to create truststore and keystore for MQ Explorer	302
6-81	Command to create the STORE.ATLANTA.QM keystore	303
6-82	Command to create the STORE.CHICAGO.QM keystore	303
6-83	Command to create the STORE.NEWYORK.QM keystore	303
6-84	Command to create the WAREHOUSE.AGENT.QM keystore	303
6-85	Command to create the ITSO.FILE.COORDINATION.QM keystore	303
6-86	Command to create the ITSO.FILE.COMMAND.QM keystore	303
6-87	Create and extract the certificate from the Atlanta store agent keystore	304
6-88	Create and extract the certificate from the Chicago store agent keystore	304
6-89	Create and extract the certificates from the NEWYORK store agent keystore	304
6-90	Create and extract the certificates from the warehouse agent keystore	304
6-91	Create and extract certificates from the MQ Explorer keystore	304
6-92	Create and extract the certificate from the STORE.ATLANTA.QM's keystore	305
6-93	Create and extract the certificate from the STORE.CHICAGO.QM's keystore	305
6-94	Create and extract the certificate from the STORE.NEWYORK.QM's keystore	305
6-95	Create and extract certificate from WAREHOUSE.AGENT.QM's keystore	305
6-96	Create and extract certificate from ITSO.FILE.COMMAND.QM's keystore	306
6-97	Create and extract certificate from ITSO.FILE.COORDINATION.QM's keystore	306
6-98	Import certificates in the Atlanta agent truststore	306
6-99	Import certificates in the Chicago agent truststore	307
6-100	Import certificates in the NEWYORK agent truststore	307
6-101	Import certificates in the warehouse agent truststore	307
6-102	Import certificate in the MQ Explorer agent truststore	308
6-103	Import the certificate in the STORE.ATLANTA.QM keystore repository	308
6-104	Import the certificate in the STORE.NEWYORK.QM keystore repository	308
6-105	Import the certificate in the STORE.NEWYORK.QM keystore repository	308
6-106	Import the certificate in the WAREHOUSE.AGENT.QM keystore repository	309
6-107	Import certificates in ITSO.FILE.COMMAND.QM keystore repository	309
6-108	Import certificates in ITSO.FILE.COORDINATION.QM keystore repository	309
6-109	Alter server connection channel for SSL configurations	310
6-110	The fteListAgents command fails with MQ error code 2397	314
7-1	The orders are placed in SOAP format over HTTP	320
7-2	The orders are placed as XML via MQ queue	321

7-3	The orders are expected in the EDI format over the MQ queue	321
7-4	Database setup script	326
7-5	inserts.sql	327
7-6	MQ setup	327
7-7	Code for PreTransformation esql node	334
7-8	esql code for the Warehouse_Protocol_Router node	336
7-9	esql code for UpdateOrderStatusWeb node	337
7-10	esql code for UpdateOrderStatusJMS node	338
7-11	esql for the node ComputeResponse esql node	345
8-1	The stylesheet used	375

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

CICS®	IBM®	System i®
DataPower device®	IMS™	System z®
DataPower®	iSeries®	Tivoli®
DB2®	Rational®	WebSphere®
developerWorks®	Redbooks®	z/OS®
FFST™	Redbooks (logo)  ®	
FileNet®	Smarter Planet™	

The following terms are trademarks of other companies:

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

This IBM® Redbooks® publication provides you with a path to demystify the complexity of adopting a service-oriented architecture (SOA) approach to integrating applications and services. With an iterative evolution of a fictitious company, which is called ITSO Enterprise, we demonstrate several scenarios about how we can implement an IBM Smart SOA approach that helps ITSO Enterprise to achieve its business goals to be a global interconnected enterprise, one step at a time.

It is not our intention to dive into the extremely technical details of every product or to tell you specific solutions for specific problems, but rather, to advise you about how to look at these problems from a business context perspective and then to provide you with a concise deployment using the IBM WebSphere® Connectivity portfolio of products to easily address them.

This book will be a reference for IT Specialists and IT Architects working on implementing Smart SOA solutions using the IBM WebSphere Connectivity portfolio of products at client sites, as well as for decision makers, IBM employees, IBM Business Partners, and IT Managers.

The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.



Virendar Solanki is a WebSphere Integration Specialist at Miracle Software Systems, Inc. He holds a Master degree in Computer Science from University of Michigan, US. He specializes in providing SOA solutions to clients using IBM products, including WebSphere ESB Server, WebSphere Process Server, WebSphere Message Broker, WebSphere MQ and WBI Adapters. He is currently focused on WebSphere ESB Server (WESB) with multiple successful client engagements.



Joao Emilio Santos Bento da Silva is a Senior IT Specialist working for IBM Brazil, acting as Delivery Center (DC) Audit and Compliance Professional focused on Middleware Infrastructure. He has 14 years of professional experience, nine dedicated to the Java platform acting as developer, architect, consultant, and WebSphere Technical Support Level 3 at companies in Latin America and the United States. Emilio holds Java certifications, including Sun Certified Enterprise Architect and WebSphere Application Server Administration since Version 5.0.



Shishir Narain is an Open Group certified Master IT Specialist having deep skills in IBM middleware products. He works in IBM Lab Services for WebSphere at India Software Labs, Gurgaon. He has 12 years of experience with multiple clients in developing solutions. He has led several end-to-end IT implementations based on SOA, including one of the largest ESB implementations in the Indian Telecom industry. He holds a Master of Technology degree from Indian Institute of Technology, Kanpur.



Matt McLarty is the Worldwide WebSphere Solution Architects leader and the leader of the global community of WebSphere Connectivity Technical Sales Specialists. Prior to joining IBM in 2007, Matt worked as an IT Director and integration architect in the banking industry for 11 years.



Rajan Kumar has been with the WebSphere Adapters product development team for more than eight years and played various roles from test engineer to a member of the core architecture team. He has led a few of the major adapter developments, such as WebSphere Adapter for IBM System i® (also known as iSeries®) and WebSphere Adapter for Enterprise Content Management. He has recently joined the Worldwide WebSphere Connectivity & Integration Sales Support team and is responsible for helping local India sales representatives to have the correct level of IBM product technical information across IBM connectivity products.



Rahul Gupta is an Advisory IT Specialist with IBM Global Technology Services in India. He is a Certified SOA Architect with six years of professional experience in IBM messaging technologies. At his current assignment, he works as a middleware consultant for various clients in North America. His core experiences are in lab testing, performance tuning, and Level 3 development for both WebSphere Message Broker and WebSphere MQ products. Rahul has been a technical speaker for messaging-related topics at various WebSphere conferences and is a recognized inventor by the IBM innovation community.



Vineet Gupta is a WebSphere Integration Specialist, having expertise in IBM Middleware products. He works in IBM Software Services for WebSphere at India Software Labs. He has seven years of IT experience, comprising various domains: Telecom, Distribution, and Content Management. He played a key role in the implementation of one of the largest ESBs in Asia. He played a critical part in designing the enterprise exception handling architecture. He holds Master of Computer Applications from Guru Jambheshwar University, Hisar.



Vasfi Gucer is a Project Leader at the IBM International Technical Support Organization. He has been with the ITSO since January 1999. He has more than 12 years of experience in the areas of systems management, networking hardware, and software on mainframe and distributed platforms. He writes extensively and teaches IBM classes worldwide on IBM products. Vasfi is also an IBM Certified Senior IT Specialist, PMP, and ITIL Expert.



Lingachary Eswarachary is a Senior WebSphere Integration Architect at IBM Software Services for WebSphere in the US. He has over 17 years of experience in the IT Industry, out of which ten years were spent working with IBM Software Group in the connectivity space. He architects, designs, and implements WebSphere Solutions and supports the end-to-end software development cycle using Smart SOA approaches. His areas of expertise include WebSphere MQ, WebSphere Message Broker, WebSphere Transformation Extender, and WebSphere IBM DataPower®. He is a Certified SOA Associate and has product certifications on other products, as well. He has worked on a few product certification development programs. He conducts training and workshops on WebSphere and related products.



Ulas Cubuk is an IT Specialist for WebSphere with IBM Software Group in Turkey and has been working in the IT industry for 11 years. Over the last two years, She has worked with many clients on SOA projects, in particular, the Business Process Management (BPM) and connectivity aspects to help clients in their particular business needs. Her current areas of expertise include BPM products and WebSphere DataPower and WebSphere ESB. In her previous roles in IBM, she worked with clients extensively on IBM Rational®, IBM System z® development, and Security and Compliance Management solutions. Before joining IBM, she focused on software quality assurance, application life-cycle management, software configuration management, and she has contributed to CMMI SCAMPI appraisals as an auditor. She has a BSc degree in Environmental Engineering and MSc degrees in Geographical Information Technologies from Middle East Technical University.



Peter Broadhurst leads a cross-product team within IBM Connectivity & Integration Development that helps our client-facing teams design SOA solutions for clients. Previously, he led the Level 3 support team for the WebSphere Application Server components that were developed in Hursley (including all JMS messaging support in the application server), as well as WebSphere MQ File Transfer edition. Before that, he worked in Development for the WebSphere MQ Version 7.0 release, and worked for a number of years in Level 3 for WebSphere MQ where he authored the popular *WebSphere MQ V6 Fundamentals* IBM Redbooks publication.

Thanks to the following people for their contributions to this project:

Tamikia Barrow, David Bennin, Richard Conway, Shari Deiana, Emma Jacobs, Stephen Smith, Debbie Willmschen
International Technical Support Organization

Brian Homewood, Ronnie Mitra
IBM UK

Kevin O'Leary
IBM US

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:
ibm.com/redbooks
- ▶ Send your comments in an email to:
redbooks@us.ibm.com
- ▶ Mail your comments to:
IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- ▶ Find us on Facebook:
<http://www.facebook.com/IBMRedbooks>
- ▶ Follow us on Twitter:
<http://twitter.com/ibmredbooks>
- ▶ Look for us on LinkedIn:
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:
<http://www.redbooks.ibm.com/rss.html>



Introduction

This IBM Redbooks publication provides you with a path to demystify the complexity of adopting a service-oriented architecture (SOA) approach to integrating applications and services. With an iterative evolution of a fictitious company, we demonstrate business scenarios that are based on real-world situations, where you can have an idea while implementing an SOA project. It is important that you understand the requirement to get business line and IT specialists together to help a company to achieve its business goals by turning it into a global interconnected enterprise, one step at time.

It is not our intention to dive into the extremely technical details of every product or to tell you specific solutions for specific problems, but rather, we advise on how we need to look at these problems from a business context perspective and provide a concise deployment using the IBM SOA family of products to address them.

This chapter summarizes the contents of this publication and introduces you to the business story that we have created to cover the most common business problems based on our several years of experience with SOA implementations worldwide.

This chapter includes the following sections:

- ▶ 1.1, “Scope” on page 2
- ▶ 1.2, “What is Smart SOA” on page 8
- ▶ 1.3, “How to use this book” on page 10
- ▶ 1.4, “Intended audience” on page 22

1.1 Scope

Because the IBM Smart SOA approach derives from IBM experiences deploying SOA at thousands of client locations, the journey from the fundamental stages to advanced implementations is a long road to document in a single book. That is why we have decided to limit its scope. In the following chapters, we cover a crucial part of what is required by an enterprise to establish the foundation for all other further phases: Reuse and Connectivity entry points.

The implementation of the scenarios uses several products to fulfill business requirements, IT constraints, and other external factors, but it does not mean that these products are your only options. We chose the products based on what has to be accomplished, but we also provide you options to address similar issues with separate architectural designs using a variety of products.

1.1.1 Business versus technology

It is never enough to say that organizations must be ready to rapidly adapt to the accelerating changes and complexity of today. We are not talking about technology itself. Constant changes in the economics and business scenarios demand a new unified approach that must bring together technology and business strategies.

To support this statement, we refer you to *The Essential CIO*, which is a biennial study based on face-to-face conversations with chief organizational officers worldwide. You can download it from [ibm.com](http://www.ibm.com) by registering first

(<http://www-935.ibm.com/services/c-suite/cio/cio-study-registration-2011.html>).

One of the key discoveries was that CIOs are now more concerned about “*how they focus their efforts and their teams on critical business objectives*”. In the next few years, we need to think about business plus technology rather than business versus technology.

The Smart SOA approach can fill this gap by establishing a foundation based on SOA services, connecting systems, reusing services, enabling automation, improving business processes slowly from basic to advanced, standardizing, and helping to guarantee simplicity and robustness in every project, no matter whether a project is small or large. IBM has both proven services and products to fill this gap effectively, based on thousands of implementations worldwide.

1.1.2 Products in scope

It is important to notice that we did not focus the contents of this book on product features, but instead, what will be a good solution based on our patterns to address certain specific business scenarios. If it does matter to you, here are the products used to implement such patterns.

WebSphere Application Server Network Deployment

WebSphere Application Server Network Deployment (ND) provides the capabilities to develop more enhanced server infrastructures. It extends the WebSphere Application Server base package and includes the following features:

- ▶ Clustering capabilities
- ▶ Edge components
- ▶ Dynamic scalability
- ▶ High availability
- ▶ Advanced management features for distributed configurations

These features become more important at larger enterprises, where applications tend to service a larger client base, and more elaborate performance and high availability requirements are in place.

WebSphere Application Server Network Deployment Edge Components provide high performance and high availability features. For example, the Load Balancer (a software load balancer) provides horizontal scalability by dispatching HTTP requests among several web server or application server nodes supporting various dispatching options and algorithms to assure high availability in high-volume environments. Using the Edge Component Load Balancer can reduce web server congestion, increase content availability, and provide scaling ability for the web server.

WebSphere Application Server Network Deployment also includes a dynamic cache service, which improves performance by caching the output of servlets, commands, Web services, and JavaServer Pages (JSP) files. This cache can be replicated to the other servers. The state of dynamic cache can be monitored with the cache monitor application. For more information about WebSphere Application Server Network Deployment V7.0, see the following web page:

<http://www.ibm.com/software/webservers/appserv/was/network/>

IBM WebSphere Services Registry and Repository

WebSphere Service Registry and Repository (WSRR) is the master metadata repository for service descriptions.

This product uses a broad definition of *service*, including these definitions:

- ▶ Traditional web services implementing Web Services Description Language (WSDL) interfaces with SOAP or HTTP bindings
- ▶ A broad range of SOA services that can be described using WSDL, XML Schema Definition (XSD), and WS-Policy decorations, but might use a range of protocols and be implemented according to a variety of programming models. For more information, see <http://www.ibm.com/software/webservers/appserv/was/network/>

As the integration point for service metadata, WebSphere Service Registry and Repository establishes a central point for finding and managing service metadata acquired from a number of sources, including service application deployments and other service metadata and endpoint registries and repositories, such as Universal Description, Discovery, and Integration (UDDI). WebSphere Service Registry and Repository is where service metadata that is scattered throughout an enterprise is brought together to provide a single, comprehensive description of a service.

When that happens, visibility is controlled, versions are managed, proposed changes are analyzed and communicated, usage is monitored, and other parts of the SOA foundation can access service metadata with the confidence that they have found the correct copy.

In this context, WebSphere Service Registry and Repository handles the metadata management aspects of operational services and provides the system for these metadata artifacts - the place where anybody looking for a catalog of all services deployed in or used by the enterprise will go first.

WebSphere Service Registry and Repository provides registry functions that support the publication of metadata about the function, requirements, and semantics of services that allow service consumers to find services or to analyze their relationships.

And it provides repository functions to store, manage, and assign a version number to service metadata.

It also supports the governance of service definitions, which provides the following functions:

- ▶ Control the access to service metadata
- ▶ Model the life cycle of service artifacts
- ▶ Manage the promotion of services through phases of their life cycles in various deployment environments
- ▶ Perform impact analysis and communicate the changes to the governed service metadata

IBM WebSphere Enterprise Service Bus

WebSphere Enterprise Service Bus (ESB) provides the capabilities of a standards-based enterprise service bus.

WebSphere ESB manages the flow of messages between service requesters and service providers. Mediation modules within the ESB handle mismatches between requesters and providers, including protocol or interaction-style, interface, and quality of service mismatches. In a Service Component Architecture (SCA)-based solution, mediation modules are a type of SCA module. The mediation modules perform a special role and therefore their characteristics differ from other components that operate at the business level.

Mediation components operate on messages exchanged between service endpoints. In contrast with regular business application components, they are concerned with the flow of the messages through the infrastructure and not only with the business content of the messages. Rather than performing business functions, they perform routing, transformation, and logging operations on the messages. The information that governs their behavior is often held in headers flowing with the business messages. The IBM SOA programming model introduces the service message object (SMO) data structure to support this pattern.

WebSphere ESB supports advanced interactions between service endpoints on three levels: broad connectivity, a spectrum of interaction models and qualities of interaction, and mediation capabilities. The product supports connectivity between endpoints through a variety of protocols and application programming interfaces (APIs):

- ▶ Java Message Service (JMS) 1.1. Applications can exploit a variety of transports, including TCP/IP, Secure Sockets Layer (SSL), HTTP, and HTTPS.
- ▶ WebSphere ESB supports Web services standards that enable applications to make use of Web service capabilities:
 - SOAP/HTTP, SOAP/JMS, WSDL 1.1
 - UDDI 3.0 Service Registry, through the WebSphere Integration Developer
 - WS-* standards, including WS-Security and WS-Atomic Transactions

Because it is built on WebSphere Application Server, WebSphere ESB can provide smooth interoperability with other products in the WebSphere portfolio, including IBM WebSphere MQ and IBM WebSphere Message Broker. It can also, with IBM WebSphere Adapters solutions, use existing application assets, as well as capture and disseminate business events.

The message clients for C/C++ and Microsoft .NET enable non-Java applications to connect to WebSphere ESB using an API similar to the JMS API.

Other features at the connectivity level perform protocol conversion between endpoints where the protocol used by the requester to dispatch requests (such as SOAP over HTTP) differs from the protocol of the service provider that is to handle those requests (such as SOAP over JMS).

IBM FileNet P8 Platform

IBM FileNet® P8 Platform is a next-generation, unified enterprise foundation for the integrated FileNet P8 products. It combines the enterprise content management reference architecture with comprehensive business process management and compliance capabilities. FileNet P8 addresses the most demanding compliance, content, and process management needs for your entire organization. It is a key element in creating an agile, adaptable Enterprise Content Management (ECM) environment necessary to support a dynamic organization that must respond quickly to change. Agile ECM solutions using IBM technologies bring together capabilities for process management, content management, regulatory compliance, and legal discovery. These flexible, extensible solutions and the ability to use these capabilities in an integrated environment can help you align key functions to your challenges and opportunities.

It includes a comprehensive set of content and process management business services that can be consumed and deployed in an SOA. It also includes multilingual system capabilities for decentralized, federated system architectures, Advanced Security Services, comprehensive auditing, and a standards-based authentication framework.

The platform supports flexible API for Java, Microsoft .NET, and XML Web services application development for a rich and interactive user experience that can be easily customized. It also supports Enterprise System Management tools, enterprise scalability, and flexible system deployment in clustered and highly available environments.

IBM WebSphere MQ

WebSphere MQ V7.0 delivers the universal messaging backbone with robust connectivity that enables flexible and reliable messaging for applications, Web services, and Web 2.0. It also delivers market-leading JMS and publish and subscribe messaging.

WebSphere MQ is the market-leading message-oriented middleware product that delivers a reliable, proven messaging backbone for almost 10,000 organizations of various sizes, spanning many industries around the world.

IBM WebSphere MQ File Transfer Edition

WebSphere MQ File Transfer Edition delivers a Managed File Transfer solution for moving files with auditability, visibility, and reliability across a broad range of platforms.

On distributed platforms, WebSphere MQ File Transfer Edition includes a copy of WebSphere MQ. This design entitles full use of the WebSphere MQ messaging functionality for traditional messaging applications, such as JMS, as well as file transfers across the same infrastructure, reducing the cost of managing and maintaining separate infrastructures for message and file data.

Trade-up licenses are available for existing clients who want to upgrade from WebSphere MQ to WebSphere MQ File Transfer Edition. Clients who upgrade can continue to use the full messaging capabilities of the copy of WebSphere MQ that is supplied with WebSphere MQ File Transfer Edition. WebSphere MQ File Transfer Edition provides a multipurpose connectivity solution for moving both files and messages.

WebSphere MQ for IBM z/OS® is a prerequisite for WebSphere MQ File Transfer Edition for z/OS. A separate license of WebSphere MQ for z/OS is required for this product when running under z/OS on the IBM System z platform.

Organizations can use WebSphere MQ File Transfer Edition to reduce and consolidate the infrastructure that is needed to move their files and messages onto a single, reliable transport that is capable of handling both kinds of traffic. Using a single reliable backbone can help

achieve these operational efficiencies by reducing the need to deploy and manage separate parallel networks for messages and files. It also enables batch-driven file-based systems to interact in an event-driven, real-time fashion by converting files into messages and vice versa. WebSphere MQ File Transfer Edition enables clients to readily and easily bring files into enterprise service buses (ESBs) to provide a modern, SOA approach to using file data and integrating it with heterogeneous environments.

IBM WebSphere MQ Advanced Message Security

IBM WebSphere MQ Advanced Message Security provides a high level of protection for sensitive data flowing through the WebSphere MQ network, while not impacting the end applications.

WebSphere MQ Advanced Message Security performs the following functions:

- ▶ Secures sensitive or high-value transactions processed by IBM WebSphere MQ
- ▶ Detects and removes rogue or unauthorized messages before they are processed by a receiving application
- ▶ Verifies that messages were not modified while in transit from queue to queue
- ▶ Protects the data not only as it flows across the network but also when it is put on a queue
- ▶ Secures existing proprietary and client-written applications for IBM WebSphere MQ

WebSphere MQ Advanced Message Security offers these features:

- ▶ IBM WebSphere MQ, Version 7.0.1 supports:
 - New interception of JMS and WebSphere MQ classes for Java

JMS: WebSphere MQ Advanced Message Security does not support the publish/subscribe domain in JMS.

- JMS 1.1 support
- Asynchronous consumer
- Message properties
- ▶ Administrative interface:
 - Command-line tools for enabling and disabling interceptors
 - Command-line tools for defining and assigning policies to queues
 - WebSphere MQ Explorer plug-in for defining and assigning policies to queues
- ▶ Independence from IBM Tivoli® software stack

IBM WebSphere Message Broker

WebSphere Message Broker is the solution for clients seeking an enterprise service bus (ESB) that provides connectivity and transformation spanning heterogeneous IT environments.

The IBM Smarter Planet™ strategy shows us how the electronic world is becoming more instrumented, more interconnected, and more intelligent. With its support for a broad range of transports, protocols, and data formats, WebSphere Message Broker performs the necessary conversions between diversely instrumented systems, which allows them to communicate with each other freely. WebSphere Message Broker enables intelligent applications and services to receive and generate data based on an interconnected infrastructure and makes smarter decisions possible.

WebSphere Message Broker offers the solution that your organization needs to transform and route business data between applications. Because it can function as a message and protocol switch, you can connect disparate applications and business data across multiple platforms. It also delivers transformation and intelligent routing capabilities for all your business information. It makes business data available exactly where you want it in the format that you need it.

WebSphere Message Broker gives you a flexible and dynamic infrastructure to simplify application integration. With its robust design, scalable architecture, high performance, and ease of use, you can implement an enterprise-wide SOA in stages.

With this flexible combination of functions, your organization is given the tools to solve your integration requirements from any starting point: a simple two-application solution or a multi-application, multi-platform design. As your business expands and the integration challenges multiply, WebSphere Message Broker can accommodate your changing business needs by offering these functions:

- ▶ Expanding your infrastructure without increasing the complexity
- ▶ Protecting your existing and ongoing investments in applications and data structures
- ▶ Seamlessly extending your connectivity capabilities

WebSphere Message Broker can accommodate many IT environments with support for an array of application and middleware technologies. In addition, it offers the following benefits:

- ▶ Exploits the industry-leading WebSphere MQ messaging infrastructure
- ▶ Supports a broad range of transport protocols (for example, integration with WebSphere MQ, JMS 1.1, HTTP, HTTPS, Web services (for example, SOAP and REST), files, Service Component Architecture (SCA), TCP/IP, and SAP Software and data formats (for example, binary, text, XML, and industry formats)
- ▶ Offers numerous ways to customize mediation (for example, route, filter, transform, and filter)
- ▶ Uses a simple programming model for connectivity and mediation, including a robust set of prebuilt patterns
- ▶ Supports a wide range of transformation options with graphical mapping, Java, ESQL, Extensible Stylesheet Language (XSL), PHP, and WebSphere Transformation Extender
- ▶ Delivers extensive administration and systems management facilities for developed solutions
- ▶ Delivers similar performance to traditional transaction processing systems
- ▶ Is available in Trial, Remote Adapter Deployment, Getting Started, and Enterprise modes

WebSphere Message Broker is the cost-effective solution for your enterprise's extensive integration and mediation needs today and helps to protect your investment into the future.

IBM WebSphere DataPower Integration Appliance XI50

The DataPower Integration Appliance XI50 model provides transport-independent transformations between binary, flat text files, and XML message formats. Visual tools are used to describe data formats, create mappings between various formats, and define message choreography. The XI50 appliance can transform binary, flat text, and other non-XML messages to help offer an innovative solution for security-rich XML enablement, ESBs, and mainframe connectivity.

In addition, the XI50 model offers the following features:

- ▶ Any-to-any transformation engine
The XI50 model can parse and transform arbitrary binary, flat text, and XML messages, including EDI, COBOL copybook, ISO 8583, CSV, ASN.1, and ebXML. Unlike approaches based on custom programming, the patented DataGlue technology of the DataPower appliance uses a fully declarative, metadata-based approach.
- ▶ Transport bridging
With support for a wide array of transport protocols, the XI50 is capable of bridging request and response flows to and from protocols, such as HTTP, HTTPS, MQ, SSL, IBM IMS™ Connect, FTP, and more.
- ▶ Integrated message-level security
The XI50 model includes mature message-level security and access control functionality. Messages can be filtered, validated, encrypted, and signed, helping to provide more secure enablement of high-value applications. Supported technologies include WS-Security, WS-Trust, Security Assertion Markup Language (SAML), and Lightweight Directory Access Protocol (LDAP).
- ▶ Lightweight message brokering:
 - Sophisticated multi-step message routing, filtering, and processing
 - Multiple synchronous and asynchronous transport protocols
 - Detailed logging and audit trail, including non-repudiation support

1.2 What is Smart SOA

Essentially, the Smart SOA approach is a set of guiding principles that IBM developed after working with several hundred clients who use IBM SOA offerings. The Smart SOA approach benefits both business and IT by extending the business value of deployment. It demands that the principles of simplicity and robustness be applied regardless of what Smart SOA style you select. It also recognizes that your needs are evolving, and while you want to make sure that you are meeting basic needs with basic projects, you also want to make sure you have room to grow when your needs evolve and become more advanced.

1.2.1 Business value of the Smart SOA approach

Smart SOA delivers distinct value with every style. Are you ready to become a globally integrated enterprise? Smart SOA business offerings from IBM can help get you there by taking an industry-focused and holistic approach to operations that provide business value at every stage of the continuum, from basic to advanced initiatives.

To become a globally integrated enterprise, start by defining your winning innovative strategy, and then implement it. Use the right tools to optimize your business processes, and then refine them continuously. Improve your ability to make better business decisions, by monitoring key industry-specific metrics. Utilize SOA to continuously align your IT and business sides of the house. And along the way, utilize the Smart SOA approach to continuously improve and optimize your business performance.

Figure 1-1 on page 9 shows the business value of the Smart SOA approach.

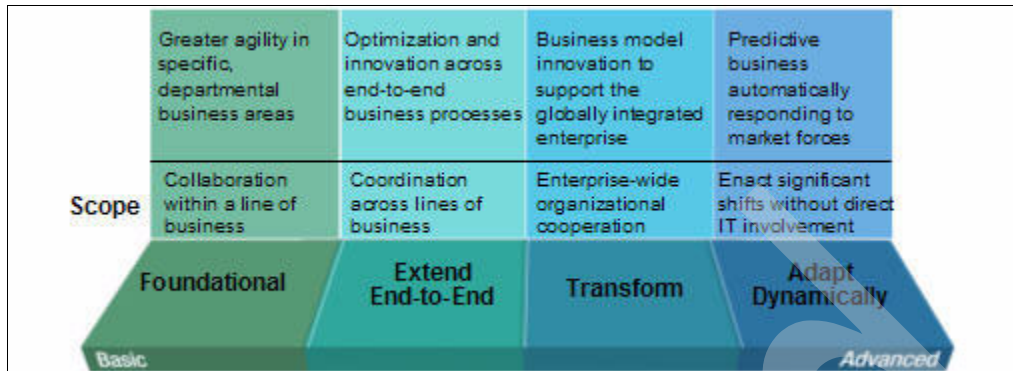


Figure 1-1 Business value of the Smart SOA approach

1.2.2 IT value of the Smart SOA approach

Technology is a major catalyst for business evolution. The Smart SOA approach not only anticipates change, it enables it with an innovative strategy for growth. Agility requires the ability to quickly and effectively respond to changes, opportunities, and threats. The Smart SOA approach is designed with agility in mind.

Figure 1-2 shows the IT value of the Smart SOA approach.



Figure 1-2 IT value of the Smart SOA approach

1.2.3 The right Smart SOA style for you

Your Smart SOA style depends on your business needs and priorities. Using the Smart SOA continuum, you can find value for all your SOA initiatives, from basic to advanced:

- ▶ Foundational: Start with focused proven, high return on investment (ROI) projects.
- ▶ Extend end-to-end: Innovate and optimize higher profile, core, end-to-end business processes for broader ROI.
- ▶ Transform: Innovate the business model by using IT for strategic advantage for ROI across the enterprise.
- ▶ Adapt dynamically: Initiate major shifts from the business side without direct IT involvement in this aspirational stage in which technology becomes invisible.

1.2.4 Learn more

There are several sources of information about IBM Smart SOA. See <http://www.ibm.com/software/solutions/soa/>.

1.3 How to use this book

This book is a story of a company that is trying to adapt to this bold new world. During this journey, the company must overcome several business challenges through the implementation of IT technologies that are available in the market to help the company succeed.

Each chapter of the book presents several steps that ITSO Enterprise has taken to complete this journey. The story evolves with the chapters and every chapter continues the story. You will notice that we skip several of the projects that were implemented by the same enterprise to give you an idea of what an enterprise can reach after changing its plan.

1.3.1 ITSO Enterprise

In this book, we step through business scenarios of a fictitious company called ITSO Enterprise. The story is based on experiences of the team working on this book, implementing or providing consulting for several clients around the world.

ITSO Enterprise is trying to stay competitive in the rapidly changing business climate of the 21st century and has significant business challenges. The old CIO has retired and a new CIO with SOA expertise has been hired. Now, the new CIO to handle several situations:

- ▶ Customers have an inconsistent experience when dealing with the company, depending on which access point they use.
- ▶ Customers are frustrated with missing, inaccurate, and stale data when dealing with the company.
- ▶ It is difficult to introduce new offers and generate new revenue due to excessive cost and long lead times.
- ▶ There is limited acceptance of the generic offers and campaigns sent out to customers.
- ▶ The company faces the potential of regulatory penalties based on new industry compliance needs, which can be traced back to inflexibility in its IT systems.
- ▶ The customer information system is closed and only updated by batch processes.
- ▶ The Customer Service Representatives (CSRs) and online systems are siloed and use incomplete customer information replicas that are updated nightly, with limited reliability.
- ▶ There is no systematic way to determine customer behavior to formulate targeted offers.
- ▶ It costs more and more money just to keep the lights on.
- ▶ Data privacy and integrity are not guaranteed.
- ▶ The internal development teams are heavily burdened, yet the culture is to code everything internally.
- ▶ Business partners want a more efficient way to interact with the company and the company wants better control of the business partners.
- ▶ Certain business partners do not perform well, but it is hard to replace them because the processes are tied to the current business partners.
- ▶ The company is dependent on the business partners instead of having control of the situation.
- ▶ All communication with the external world is manually performed by either email, phone calls, or old forms.

ITSO Enterprise is now turning to becoming an SOA company. ITSO Enterprise knows that SOA is the best way to move forward instead of deploying siloed projects that do not cover its future needs and sometimes can be an obstacle.

1.3.2 Contents

ITSO Enterprise realized that it is time for a change. Several problems that in the past did not directly affect the company are now becoming real financial losses. We have organized this book to address the evolution to an SOA Foundation implementing connectivity solution patterns.

Chapter 2: Service Enablement pattern

Chapter 2, “Service Enablement pattern” on page 23 is the beginning of the journey. The first SOA service is about to be created to address a problem that has persisted for a long time.

Business context

Since ITSO Enterprise released its web portal, customers have been presented with inaccurate, stale data that was provided by the Customer Service Representatives (CSR). Both customer channels are siloed in their own departments and maintain separate copies of customer data. The data is updated once a day. Figure 1-3 explains the update.

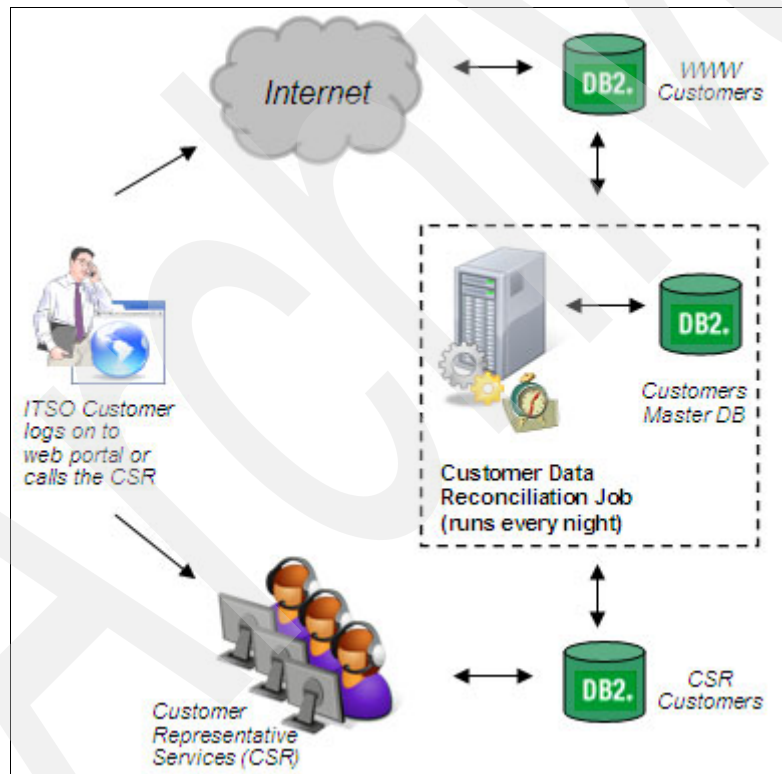


Figure 1-3 ITSO Enterprise customer channels and stale data

Smart SOA solution

As a solution for this problem, an SOA service will be created and deployed as an enterprise service bus (ESB) flow. All applications managing or requesting customer services will centralize requests. Updates on both databases will be done every time that the service is called, providing the synchronization of the data in real time (Figure 1-4 on page 12).

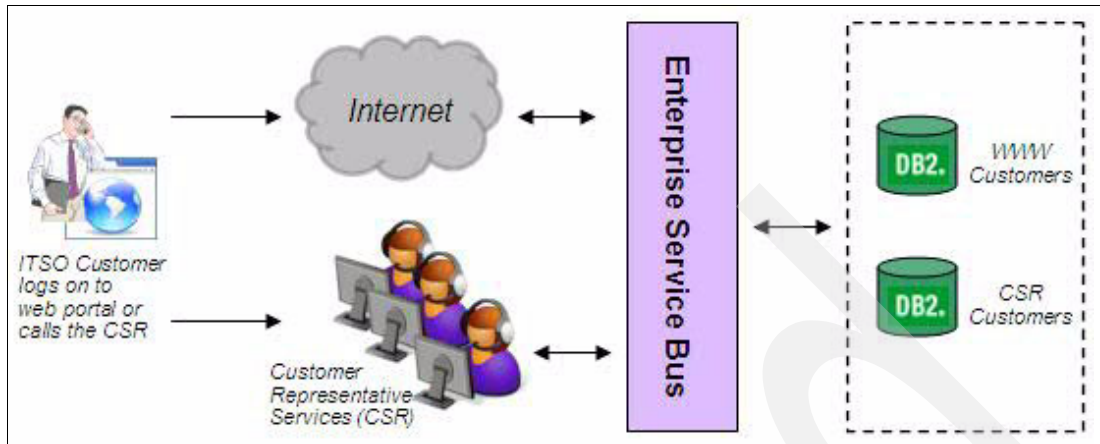


Figure 1-4 Service Enablement

Connectivity pattern

The Service Enablement pattern will be implemented to provide one entry point for all applications that work with customer information. We implement a mediation flow that manages customer databases in a single transaction (Figure 1-5).

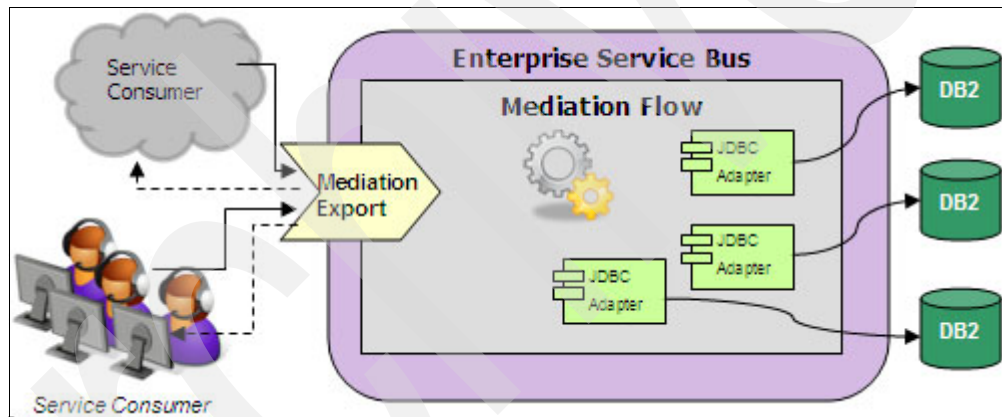


Figure 1-5 Service Enablement

Products: Figure 1-5 demonstrates our implementation of the pattern. You can call other external components, such as MQ queues or web services.

Chapter 3: Service Enablement pattern for ECM system

In Chapter 3, “Service Enablement pattern: Enterprise Content Management System” on page 75, we describe building a Service Enablement solution on WebSphere Enterprise Service Bus using IBM WebSphere Adapter for Enterprise Content Management (ECM).

Business context

After ITSO Enterprise has spent many years in the local market, the company thought it was time to globalize its business. However, because each country has its own laws and legal processes, ITSO Enterprise wanted to establish a standard mechanism within its organization to manage its suppliers’ legal contract documents. It involves storing scanned physical

contract documents that have been signed between ITSO Enterprise and its global suppliers and reviewing them on a periodic basis for any renewals, rejections, or approvals.

Also, because ITSO Enterprise is a leader in implementing many environmental policies, it needed to comply with the Paperwork Reduction Act to reduce paperwork by storing documents on IBM FileNet and making the documents available on demand to only selected teams or legal officers. The highly confidential documents contain crucial information, such as agreed-on price, contract terms, length of contract, and so on (Figure 1-6).

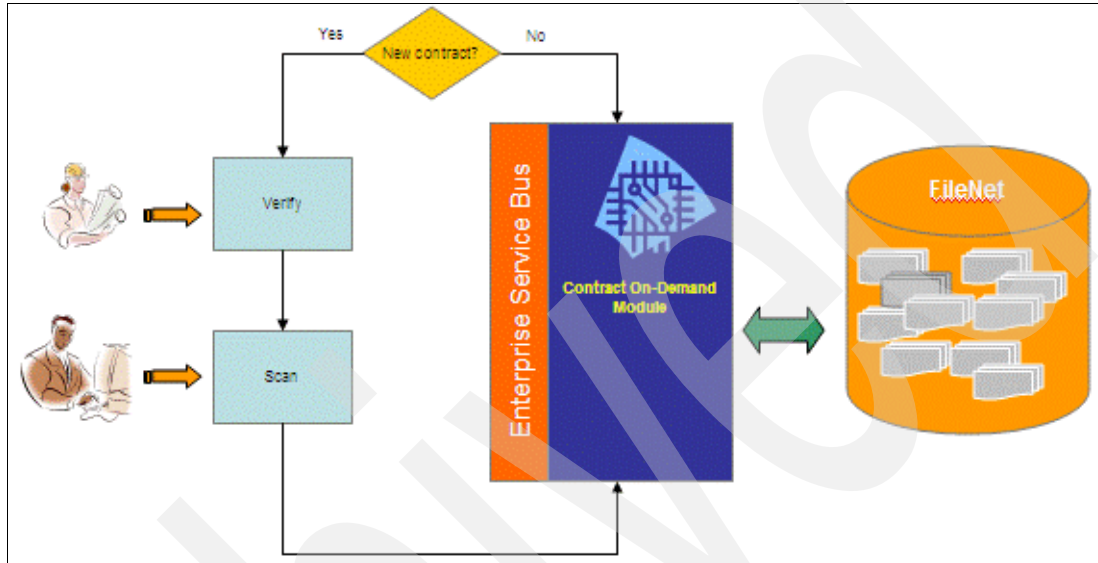


Figure 1-6 Business context for Service Enablement of ECM

Smart SOA solution

To enable this capability, ITSO Enterprise built an internal web application that uses WebSphere Adapter for Enterprise Content Management and exposes FileNet content as an appropriate business service per the SOA model.

Connectivity pattern

With WebSphere Adapter for Enterprise Content Management, you can create integrated processes that include the exchange of information with a Content Management Repository, without special coding. The adapter has been built using the open source specification for content repositories called Content Management Interoperability Services (CMIS).

Chapter 4: Service Governance pattern

Based on the success of accessing a single service across departments with the customer information service, as described in Chapter 2, “Service Enablement pattern” on page 23, now, ITSO Enterprise plans to address other problems that an SOA approach can solve, such as the lack of integration of multiple services across all departments.

Business context

After placing an order, if a customer wants to know its status, the customer can either check on the web portal or call a Customer Sales Representatives (CSRs). The CSRs do not have direct access to the ITSO Enterprise ordering system. To provide the information that the customer wants, the CSR puts the call on hold and phones the ITSO Enterprise ordering team to query the order for the given customer, as described in Figure 1-7 on page 14. Customers get frustrated when they are put on hold.

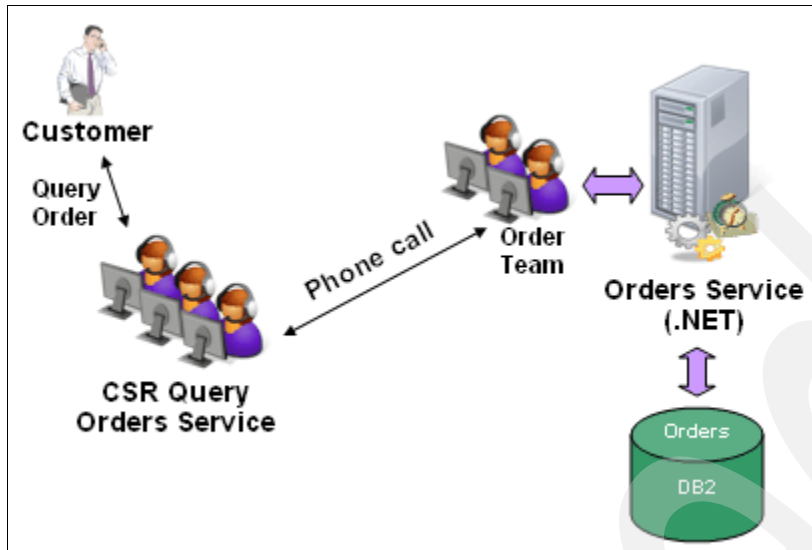


Figure 1-7 ITSO Enterprise: CSR providing order information

ITSO Enterprise can use the same solution for the previous project, but the Ordering team is concerned about enabling a service that can be used by anyone in the company without control.

Smart SOA solution

IBM helps ITSO Enterprise to solve the problem by showing the company how to reuse the ordering system to enforce that only controlled requests can be made to access it.

Service Reuse will be implemented again, but this time, a Service Governance solution is part of it. A service-level agreement (SLA) can make sure that only previously registered requesters can access data.

Connectivity pattern

ITSO Enterprise knows how to reuse services, but like any other company, it must control the life cycle of the services. Letting this management spread across multiple teams in a decentralized manner without an effective solution might bring future complications. Deploying a services registry and repository allows companies to manage and govern services. For this particular problem, an SLA is implemented to be used inside a service gateway flow that only responds to requesters that are compliant. After implementation, the solution looks like Figure 1-8 on page 15.

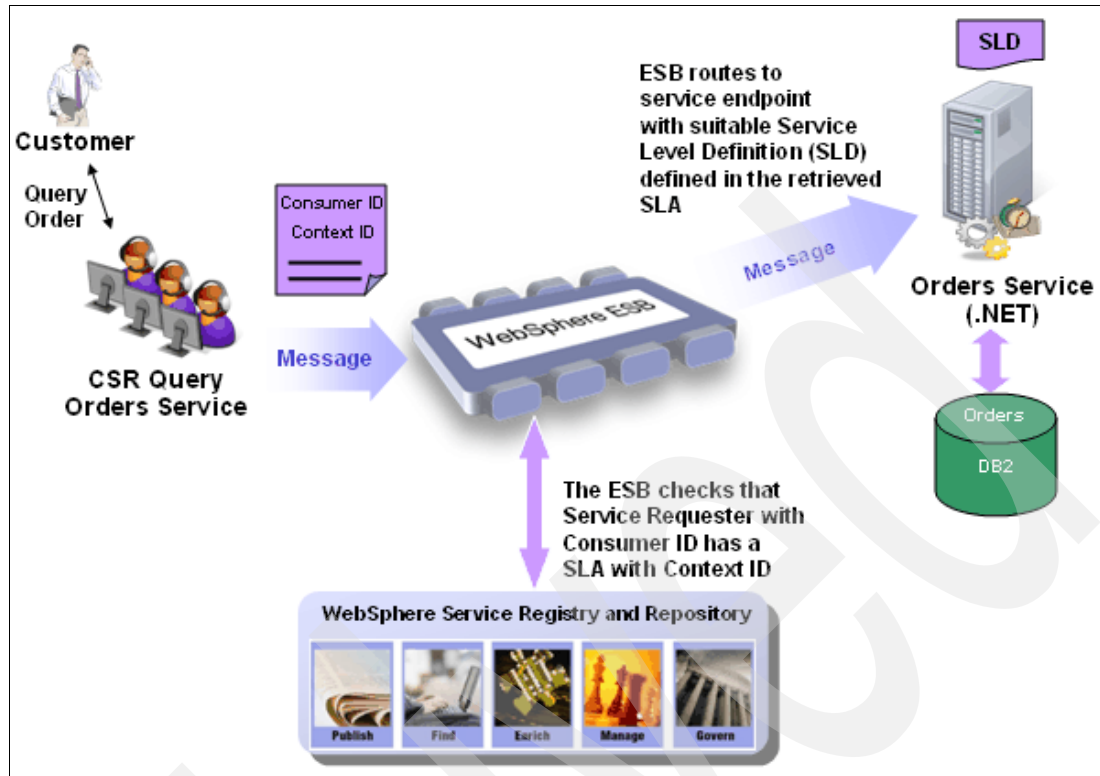


Figure 1-8 Service Governance with SLA

Chapter 5: Message Privacy Enforcement pattern

Chapter 5, "Message Privacy Enforcement pattern" on page 201 introduces a security policy technical requirement to be implemented within an existing messaging system that holds sensitive information.

Business context

Changes do not come from only inside the company. The PCI-DSS is now a mandatory requirement for the ITSO Enterprise. Processes and applications must be created or changed to comply with the security policy standards and requirements.

One of the changes that has been identified by the IT Architect in charge of the project is the communication between the point-of-sale (POS) and the data center that processes all credit card authorizations.

All transactions are placed on an asynchronous messaging system. Every store has its own messaging server. Messages are taken from the queue by the authorization server application and sent to the proper authorization company. All messages are exchanged using secure protocols, and authentication is in place, as shown in Figure 1-9 on page 16. The problem is that messages are being stored in the queues without encryption. Encryption of the credit card number is mandatory when stored.

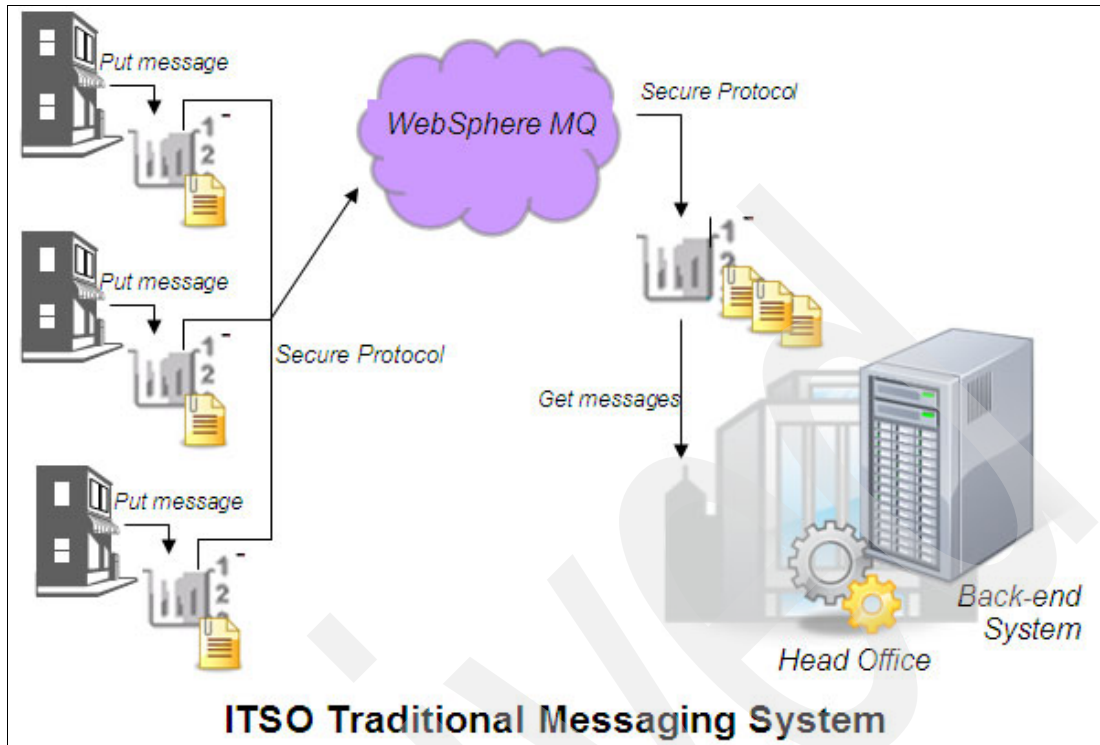


Figure 1-9 Authorization Service Messaging System

ITSO Enterprise has to comply with the requirement within a period of time to avoid penalties. Developers are willing to update all PoS applications and the server process that makes the call to the authorization services to encrypt and decrypt messages containing credit card information. It will take too much time to develop, test, and after deployment, maintain an in-house application because the developers are already overburdened.

Smart SOA solution

The solution is to enable a mechanism that encrypts messages entering the queues and decrypting the messages before they leave the queue, as you can see in Figure 1-10.

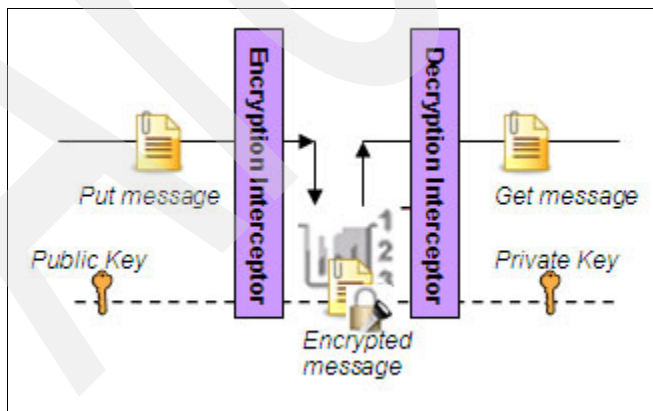


Figure 1-10 Message encryption

IBM WebSphere MQ Advanced Message Security replaces IBM WebSphere MQ Extended Security Edition with a simple architecture that can be installed and configured on the existing solution to enable privacy for the messages that are at rest on the queues.

Connectivity pattern

Because IBM WebSphere MQ Advanced Message Security will be installed, no changes to the existing PoS application code will be required.

In an additional effort to keep track of all messages being exchanged, due to auditability requirements and the implementation of a new authorization SOA service, a service gateway will be updated to be a security gateway between the stores and the authorization service. Figure 1-11 demonstrates how the solution.

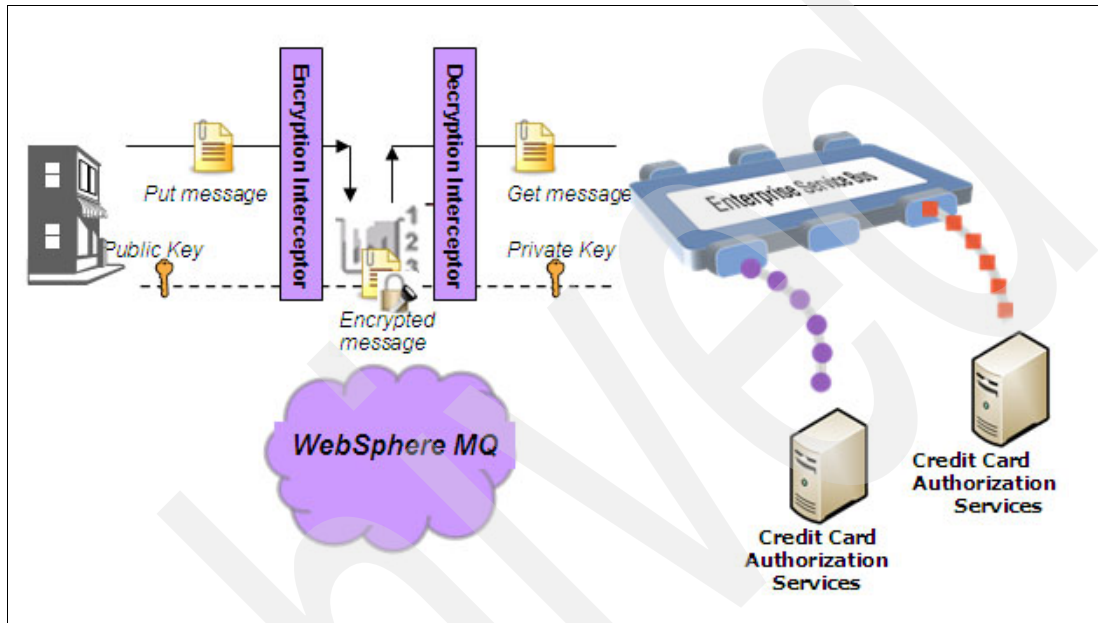


Figure 1-11 Enforce message privacy

Chapter 6: Enterprise Messaging and File Backbone pattern

In Chapter 6, “Enterprise Messaging and File Backbone pattern” on page 233, we describe a common scenario of file transfer and processing.

Business context

A new age is now starting for ITSO Enterprise. The business results of the previous successful Smart SOA projects now drive changes across the organization. Security and reliability policies are defined throughout the company and those policies are mandated in all projects.

ITSO Enterprise is a centennial company (over 100 years old), with several stores around the country and a centralized warehouse, that negotiates with various suppliers and ensure that store inventories do not drop beneath the low inventory threshold.

Store systems are scheduled to send an actual daily inventory file to the warehouse, so that the warehouse can synchronize the file with the centralized organizational inventory. These files are sent over non-secure, unreliable, and primitive FTP protocol with a risk of files being transferred incorrectly or lost. Figure 1-12 on page 18 provides an overview of the current mechanism of how stores transfer their inventory to the warehouse.

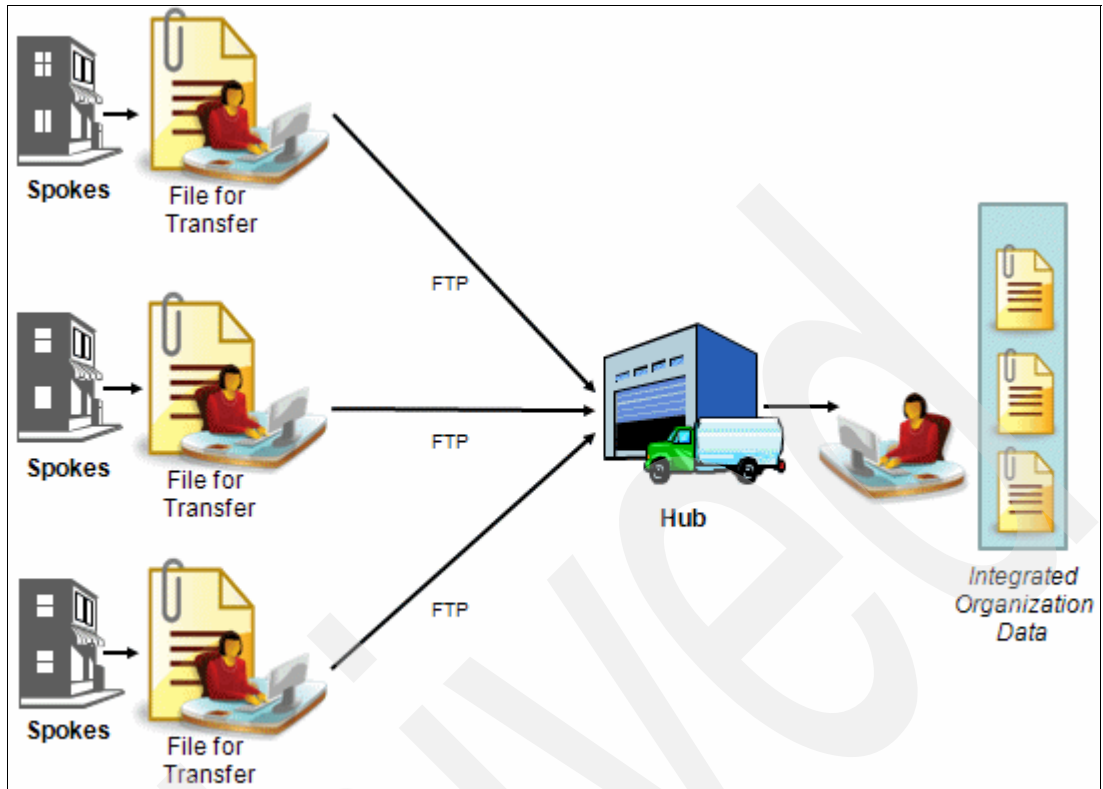


Figure 1-12 Unmanaged file transfer currently deployed in the ITSO Enterprise company

ITSO Enterprise has had problems with old, native FTP systems for a long time. Store inventories and the warehouse systems show unsynchronized quantities, which often leads to supply issues. If the file transfer fails at the scheduled time, the current systems are unable to detect the problem and restart the file transfer. Also, ITSO Enterprise is unable to audit the past file transfers from the stores to the warehouse.

To augment the existing problems, the staff in the stores is not technical. The non-technical staff members are unable to resolve the problems caused by using native file transfer techniques, hence causing delays while processing organizational inventory and supplies.

Smart SOA solution

ITSO Enterprise decided to implement a Managed File Transfer (MFT) solution. ITSO Enterprise reviewed the Managed File Transfer technology and decided to implement several of the features in the Smart SOA Managed File Transfer domain, as shown in Figure 1-13.



Figure 1-13 Managed File Transfer features

Connectivity pattern

Before implementation was started for a Managed File Transfer solution, an Enterprise Messaging and File Backbone integration pattern was suggested to ITSO Enterprise, as shown in Figure 1-14.

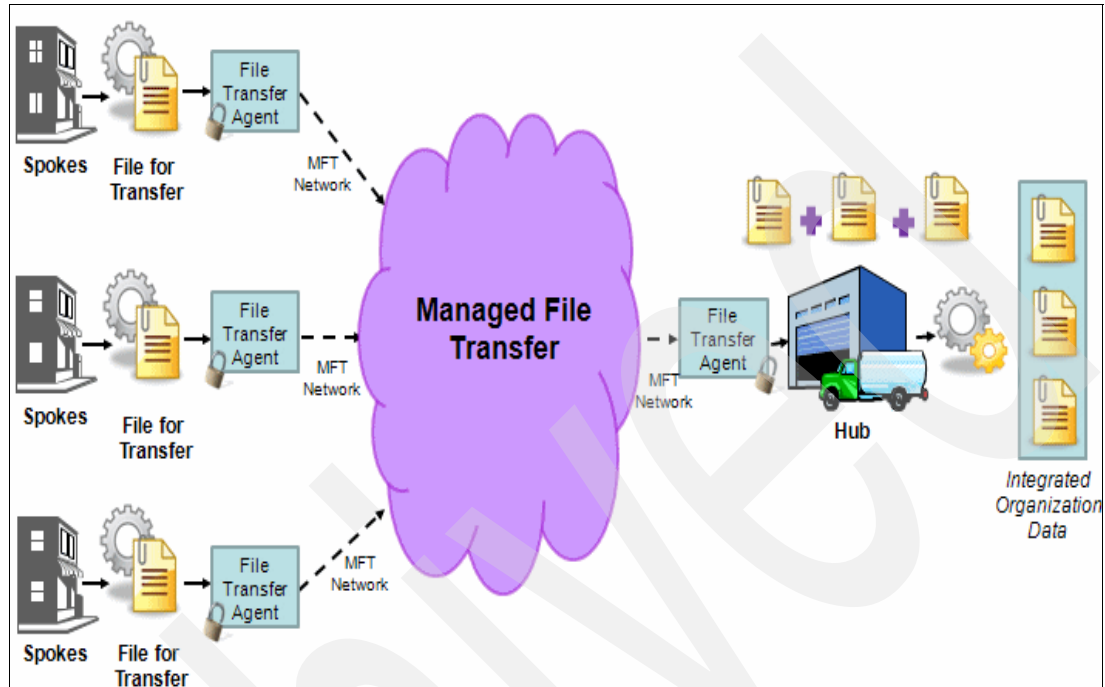


Figure 1-14 Enterprise Messaging and File Backbone pattern

Chapter 7: Evolve to SOA pattern

Chapter 7, “Evolve to SOA pattern” on page 317 introduces an SOA scenario that shows how ESB can be used to meet the challenges of connectivity with the suppliers.

Business context

The ITSO Enterprise IT Department has identified the following challenges with respect to the integration of the warehouse with the widget suppliers:

- ▶ The suppliers require various message formats (EDI, XML, and SOAP) over separate protocols, such as FTP, JMS, and SOAP. The current program works with both the transformation and protocol variations. Due to these variations, it is difficult to maintain the program, because it must be consistently upgraded to meet the supplier’s requirements.
- ▶ Order tracking with the suppliers is manual and takes a lot of time for the ITSO Enterprise staff. An online automated process will be faster and cost-effective.
- ▶ Any change to the supplier endpoint forces a change in the Order program. The various suppliers support separate messaging and technology protocol. Keeping the Order application up-to-date strains the ITSO Enterprise IT resources, because even a single day of outage means unsatisfied widget customers.

The recent success with IT has spurred the management to improve the supplier integration by implementing these methods:

- ▶ Introducing Enterprise Service Bus (ESB) infrastructure
- ▶ Using proven products to meet the integration challenges and introduce agility

Smart SOA solution

The solution utilizes IBM WebSphere Message Broker to realize the ESB. The built-in nodes of WebSphere Message Broker for file processing, message transformation, and transports takes out the technical challenges and allows ITSO Enterprise developers to focus on the business needs. Another important aspect of the SOA solution is that it works with the existing Inv ITSO Enterprise inventory software.

WebSphere Message Broker also enables ITSO Enterprise to expose services that can accept the Order status updates.

Connectivity pattern

An ESB is an architectural pattern that supports virtualization and the management of service interactions between communicating participants. It acts as an intermediary for connectivity services between service providers and requesters in an SOA. It is a flexible connectivity framework that facilitates reliable and secure system integration while reducing the number, size, and complexity of application interfaces.

Rather than interacting directly, services communicate through a service connectivity framework (the ESB) that provides virtualization and management features that implement and extend the core definition of SOA (Figure 1-15).



Figure 1-15 Evolve to SOA

Chapter 8: Hybrid Bus pattern

In Chapter 8, “Hybrid Bus pattern” on page 353, ITSO Enterprise reuses the current infrastructure to both consume and expose services from and to outside parties, complying with all IT policies and security standards.

Business context

As part of the organization’s commitment to continually improve its IT business processes, ITSO Enterprise decided to expose the ordering Web service to business partners and customers to extend its business and secure supplier-specific work orders and the order status messages that were received from the suppliers.

The IT Security Department enforced the following security policies:

- ▶ All incoming requests must be authenticated. (Although customers can choose separate authentication mechanisms, WS-Security username token is the most commonly used method.)
- ▶ All outgoing order requests must be secured.
- ▶ The transport layer must be protected.
- ▶ Sensitive information must be encrypted at the message level.

Smart SOA solution

The solution utilizes IBM WebSphere DataPower, an ESB Appliance, as the XML Security Gateway to meet the organization's security needs. The order service is securely exposed to external partners via the ESB Appliance. Suppliers connect to ITSO Enterprise securely over DataPower to send order requests and order statuses, as shown in Figure 1-16.

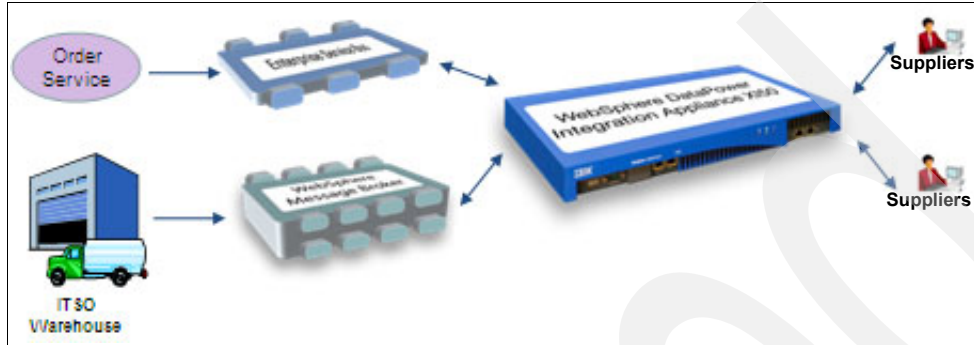


Figure 1-16 Hybrid Bus pattern

Connectivity pattern

ITSO Enterprise will implement the Hybrid Bus pattern to achieve greater efficiency and flexibility. As shown in Figure 1-17, the pattern extends WebSphere Enterprise Service Bus and WebSphere Message Broker to secure partners' interaction with ITSO Enterprise by integrating them with WebSphere DataPower. DataPower is deployed in the DMZ, and WebSphere Enterprise Service Bus and WebSphere Message Broker are deployed in the trusted zone.

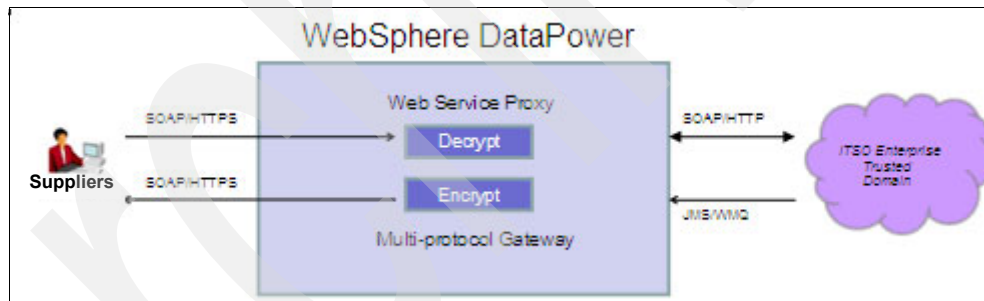


Figure 1-17 Smart SOA solution

Chapter 9: Conclusion

Chapter 9, "Conclusion" on page 395 provides you a summary of the benefits that were achieved in each chapter, either on the business or IT side. What benefits can we achieve from implementing the Smart SOA approach at ITSO Enterprise? We think Smart SOA and the ITSO Enterprise scenario can help answer questions, such as:

- ▶ What can be done next?
- ▶ How can we identify other business scenarios?
- ▶ Does that fit a company my size?
- ▶ What differentiates Smart SOA from other approaches?

In this chapter, you can also find additional resources for SOA projects.

1.3.3 Connectivity patterns that are used in the book chapters

Table 1-1 lists the connectivity patterns that we have used in this book and the corresponding scenario chapters.

Table 1-1 Connectivity patterns that are used in the book chapters

Chapter	Connectivity pattern
Chapter 2	Service Enablement
Chapter 3	Service Enablement of the Enterprise Content Management System
Chapter 4	Service Reuse and Governance
Chapter 5	Message Privacy Enforcement
Chapter 6	Enterprise Messaging and File Backbone
Chapter 7	Evolve to SOA
Chapter 8	Hybrid Bus

1.4 Intended audience

If you want a better view of the Smart SOA approach and how it is implemented to solve real business problems, this book is right for you. If you are from the business side, for example, you design processes or manage an area that is willing to get ready for future endeavors of this ever-evolving world of rapid changes, we can help you to identify the challenges that already have a defined solution defined and we can support you to achieve your goals. Moreover, we might help you think differently, looking for smarter solutions for your current business problems.

If you are on the IT side, this book is a great opportunity for you to realize that simplicity and robustness can work together. IT solutions must work hard to be invisible for the business, and there is no better way than establishing an SOA foundation using well-proven implementations to solve simple to complex problems.

We hope you have time to read the entire book, but if you want to focus your attention on aspects specific to business line or specific to IT implementations, we have structured the chapters to fit these needs as well, separating what is really important for separate interests. For instance, an IT Architect might want to see a diagram of the solution. An Integration Specialist wants to understand how mediation flows have been implemented. We created this book for all of us: Business experts, Pre-sales IT Architects, Integration Developers, Infrastructure Administrators, and all other interested parties.

Service Enablement pattern

This chapter describes the Service Enablement pattern using the Smart Service-Oriented Architecture (SOA) approach.

In this chapter, we introduce IBM WebSphere Enterprise Service Bus within the context of ITSO Enterprise, which is facing challenges in the business processes and connectivity of various applications and systems.

This chapter has the following sections:

- ▶ 2.1, “Introduction to the business scenario” on page 24
- ▶ 2.2, “Architectural diagram” on page 26
- ▶ 2.3, “Benefits” on page 26
- ▶ 2.4, “Technical implementation” on page 27
- ▶ 2.5, “Summary” on page 73

2.1 Introduction to the business scenario

ITSO Enterprise has a CSR (Customer Service Representatives) system, which is accessed by the sales representatives to add, update, or find customer details in the CSR (IBM DB2®) database system. ITSO Enterprise also has a web portal system, which allows customers to create and update their profiles. Customers can place an online order using the web portal system. The ITSO Enterprise WWW (DB2) database system stores all customer details and placed orders that come through the web portal system.

A reconciliation program runs nightly to synchronize both the CSR and WWW databases to the Master database (a DB2 database), based on the update flag in the CSR and WWW databases, as shown in Figure 2-1.

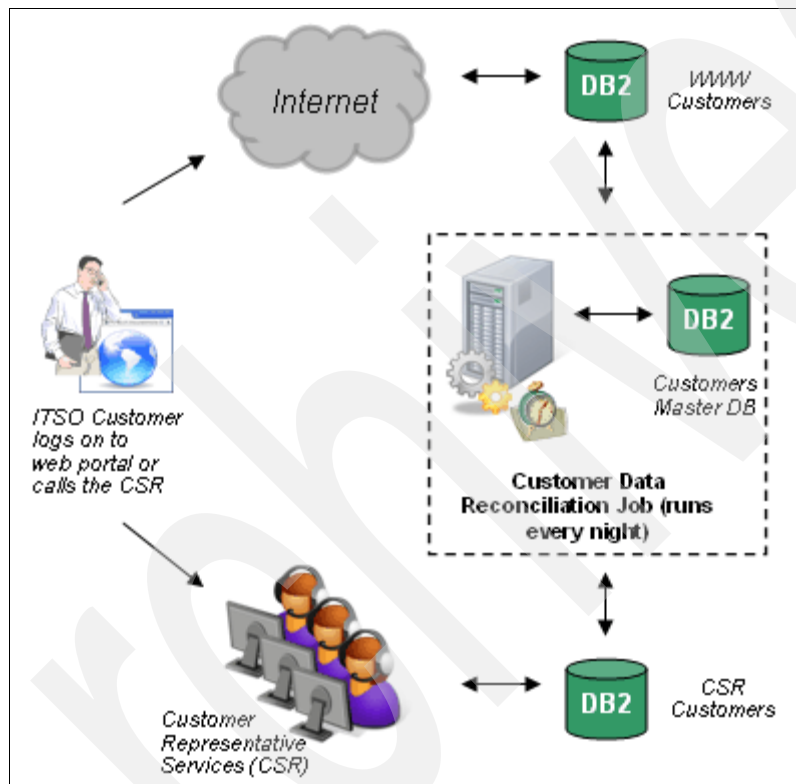


Figure 2-1 ITSO Enterprise infrastructure

2.1.1 Scenario problem

A customer logs in to the ITSO Enterprise web portal to place an order. The customer has not placed an order in a long time and the customer realizes that it needs to change its address.

"That's fine, I will just change my address before I place the order."

After the customer changes the address, the customer receives a response from the website that it will take 24 hours before the address change goes into effect. A bit frustrating, but the customer knows ITSO Enterprise has good customer service, so the customer phones the local branch to place the order.

The CSR who answers the phone checks its own CSR system to see whether the customer's new address has been updated and sees that it has not.

“No problem, this happens all the time. I will talk to the orders team and place the order manually, then give you a call back to confirm. The order will be placed tomorrow. May I just get an up-to-date contact number?”

The CSR updates its system with the new number. The CSR places the order manually with the Orders team and then calls the customer back to tell the customer that the order will be placed tomorrow. The next day, the customer receives notification that the order has been placed and an e-mail stating that the address update is now complete.

The customer decides to order more widgets. At least, this time, the customer can order them online. The customer logs back into the ITSO Enterprise web portal, checks to see if the address has been updated, and finds that the address is the same as it was yesterday. Only the telephone number has been updated. Consequently, the customer calls back to customer service and cancels the order.

2.1.2 Solution

To resolve this problem, ITSO Enterprise uses the Smart SOA approach using IBM WebSphere Enterprise Service Bus (WebSphere ESB) to integrate all three of the systems that hold the data: Master (DB2), WWW (DB2), and CSR (DB2), as described in Figure 2-2.

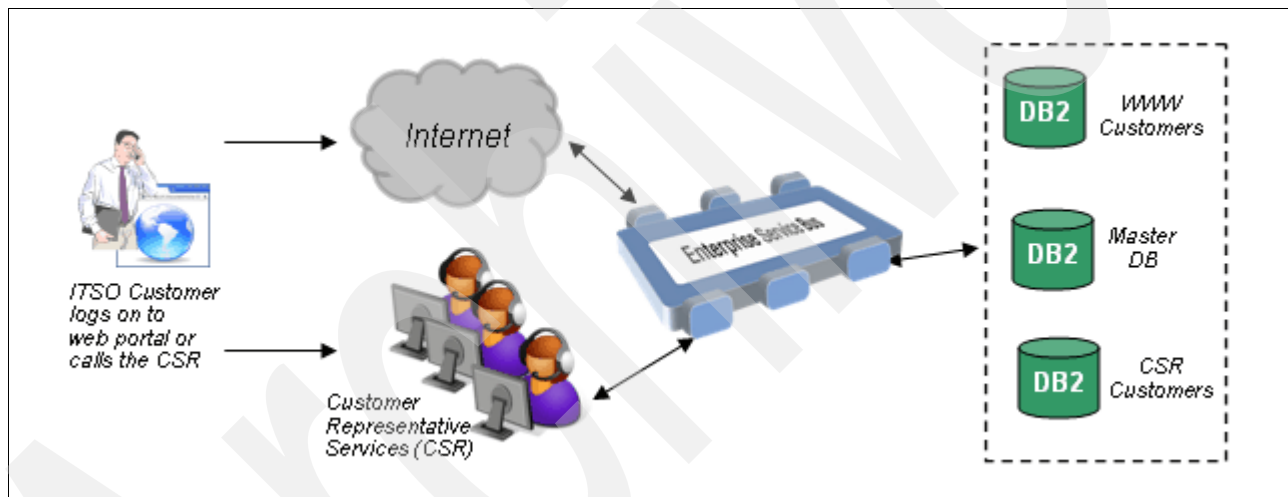


Figure 2-2 IBM WebSphere ESB solution for ITSO Enterprise infrastructure

ITSO Enterprise uses the transactional capability of IBM WebSphere ESB, together with its capabilities for communicating with both databases and systems, and exposes the service as a transactional SOAP/HTTP service. Both the web portal (.Net Application) and CSR application (Java thick client application) are updated to call the same service when adding, updating, or finding customers.

The CSR application and web portal application both retain their own local databases in this approach, so that the customer's original concerns over the capacity of the master customer information system to handle the volume of read-access transactions are not a problem. In the previous solution, both the CSR application and web portal application were using the master customer information system for read access transactions, which can cause performance problems during peak demands. In the current system, each application runs its own local database, so this conflict is not an issue.

There is no change to the user interface that the CSR team uses, so no additional training is required.

The next time that the customer updates its address, the customer receives immediate confirmation that the address has been updated. The customer phones the branch to check, because of the customer's previous experience, and the CSR system is now in sync, too.

2.2 Architectural diagram

In this scenario, we use the following products:

- ▶ IBM Integration Designer V7.5
- ▶ IBM WebSphere ESB Server V7.5
- ▶ IBM DB2 database V9.7

Figure 2-3 shows an architectural diagram of ITSO Enterprise with an IBM WebSphere ESB solution.

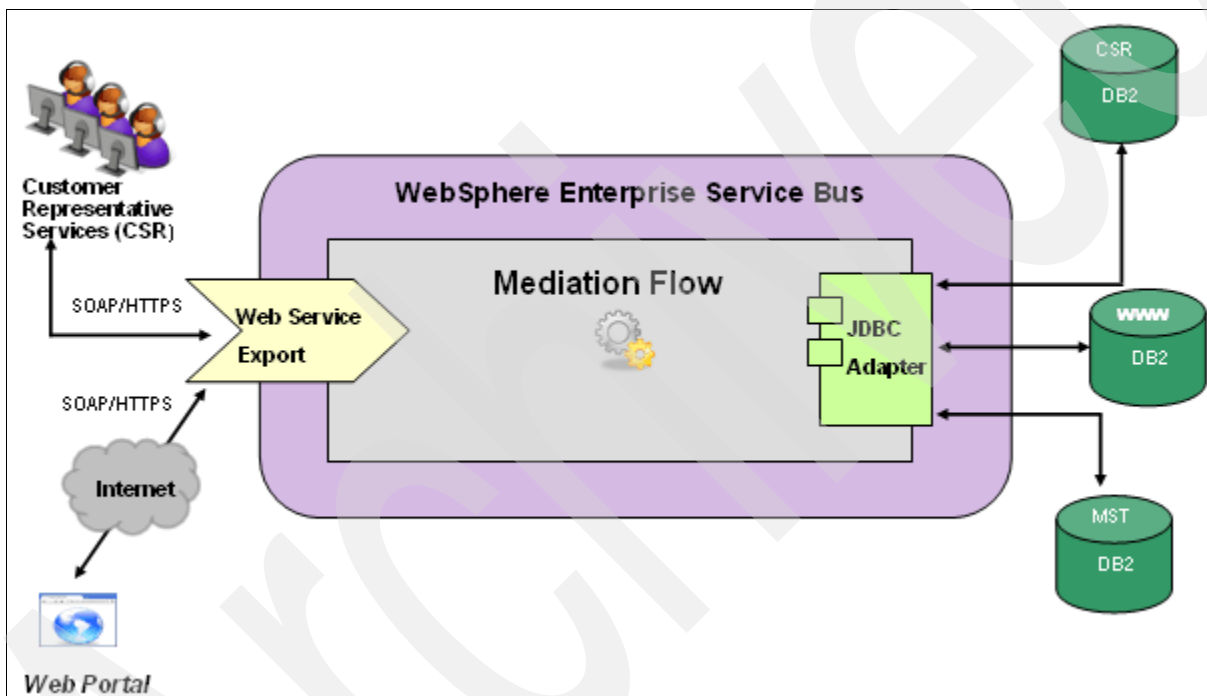


Figure 2-3 An architectural diagram with a WebSphere ESB solution for ITSO Enterprise

Multiple databases: ITSO Enterprise keeps three DB2 databases with redundant customer details to maintain tightly coupled applications and report generation systems with each of the DB2 databases. ITSO Enterprise does not want to invest time and money to modify its existing applications. Hence, ITSO Enterprise makes a business decision to exist with all three DB2 database instances.

2.3 Benefits

In general, service integration refers to integrating separate applications running on separate platforms and using separate protocols.

In our scenario, ITSO Enterprise uses CSR (a Java thick client application) and web portal (a .NET application), both communicating to a DB2 database system as a back end. The reconciliation program runs daily once at night time to synchronize both the CSR and WWW databases to the Master database, so if the customer wants to make any update on the web, it takes 24 hours for the update to be active.

To overcome this business problem, ITSO Enterprise needs to integrate the CSR and web portal applications with real-time transaction capability, so that customers can update any data at any time on the web.

IBM offers a few products that support ESB functionalities. Based upon business needs, customers can choose any of the ESB products to implement service integration in their IT environments.

In this scenario, we use IBM WebSphere Enterprise Service Bus as the ESB product to integrate the CSR application, web portal application, and DB2 database system.

2.3.1 Enterprise Service Bus features using WebSphere ESB

IBM WebSphere ESB is a platform-independent ESB, which can be used to integrate disparate applications, to transform separate types of format data between any type of applications, using separate communication protocols or distribution methods.

Using this approach, we place a single logical node into the middle of the enterprise infrastructure and broker every transaction flowing through a single ESB node. WebSphere ESB is made up of service consumers on one side, service providers on the other side, and mediations in the middle, which allow them to communicate.

2.4 Technical implementation

To implement the scenario, we use the following steps:

1. Import the ITSO library into the IBM Integration Designer workspace.
2. Build a mediation module for Add/Update/Find customers.
3. Assemble the mediation module.
4. Create a Java Database Connectivity (JDBC) outbound adapter.
5. Implement the mediation module.
6. Test the mediation module.

2.4.1 Importing the ITSO library into the IBM Integration Designer workspace

The ITSO library contains all artifacts, such as the business objects and interfaces that relate to the customer information service.

From the Integration Designer top-level menu, follow these steps:

Select **File** → **Import**.

7. In the Import dialog box, select the **Project Interchange** format from the Other folder, as shown in Figure 2-4 on page 28.

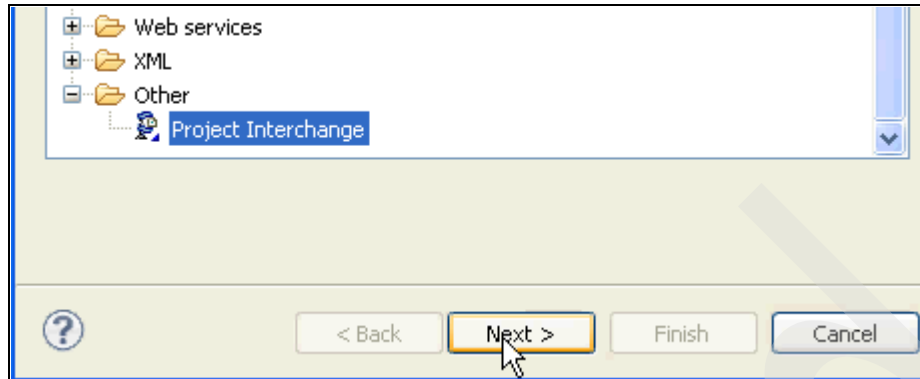


Figure 2-4 Import dialog box

8. On the next window, follow these steps:
 - a. Click **Browse** next to the “From zip file” field.
 - b. Browse to the location where you have stored the `ITS0_Lib.zip` and click **Open**.
 - c. Select the library with a check mark and click **Finish** (Figure 2-5 on page 29).

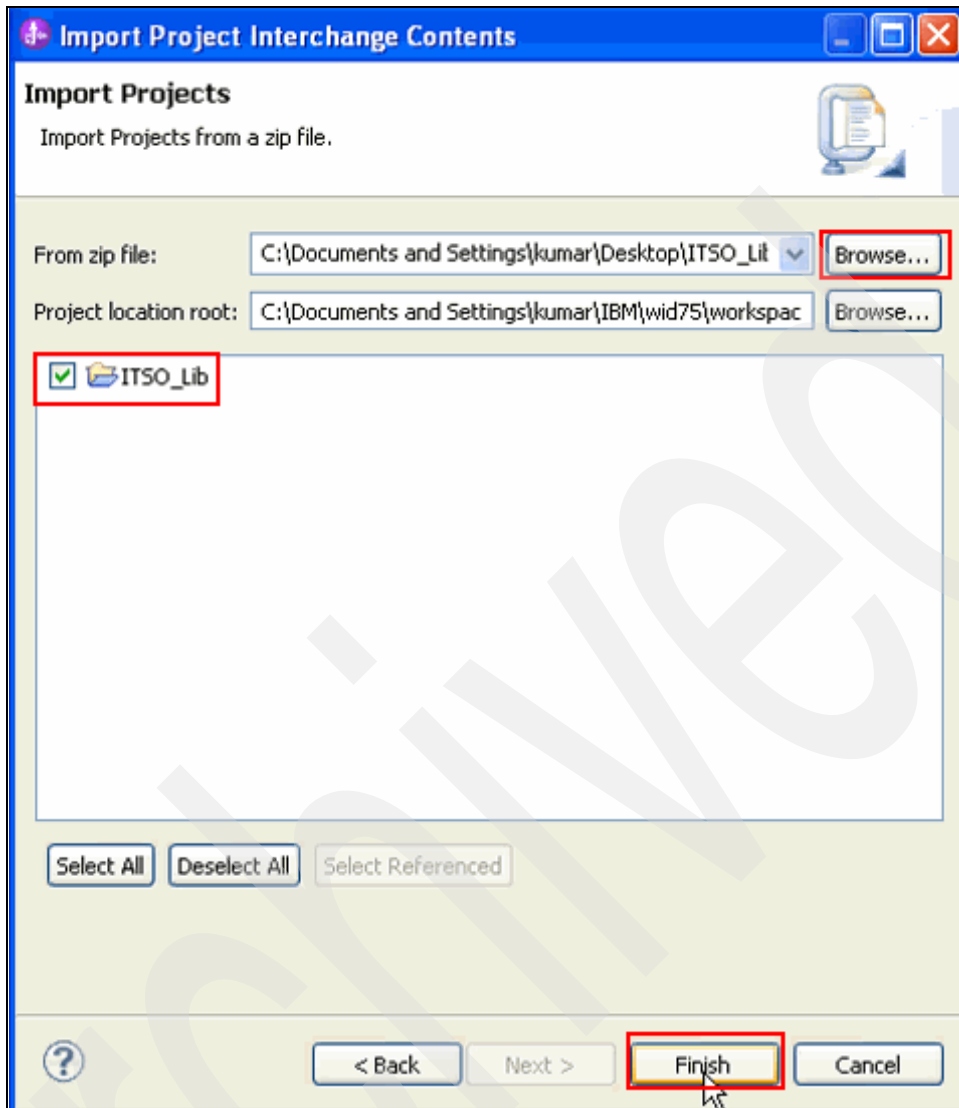


Figure 2-5 Import Project Interchange Contents dialog

9. In the Business Integration view, expand the library project tree, as shown in Figure 2-6.

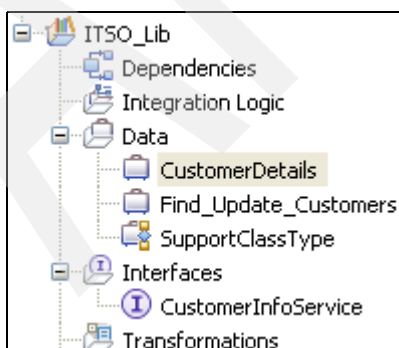


Figure 2-6 Library project view

2.4.2 Building the mediation module for Add/Update/Find customers

We describe how to create the mediation module:

1. In the Business Integration view, right-click in the white space and select **New** → **Mediation Module** (Figure 2-7).

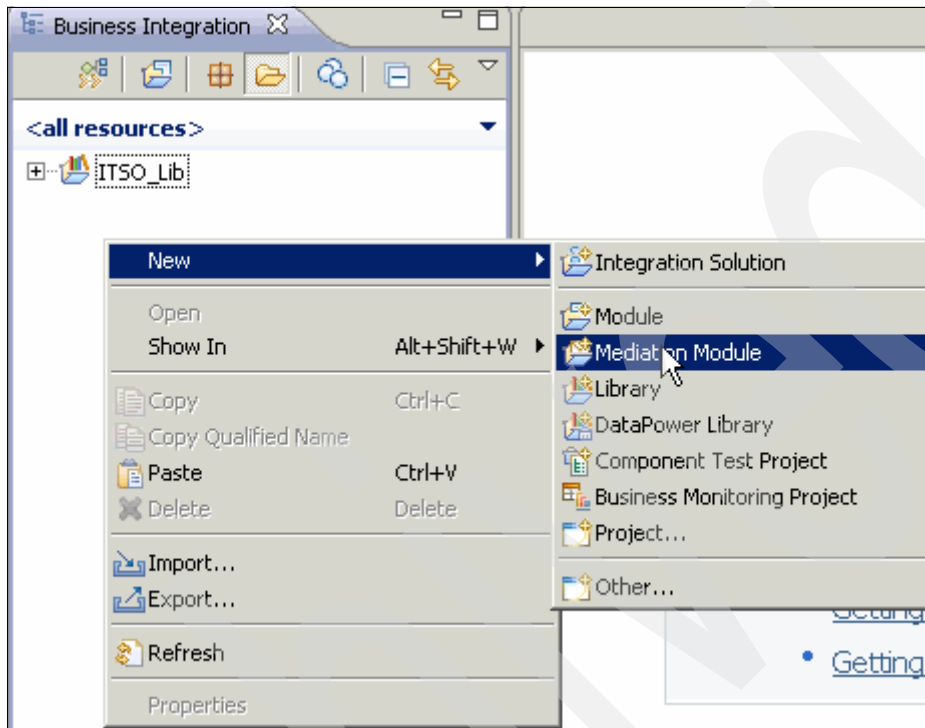


Figure 2-7 Create the mediation module

2. Enter the Module name as `ITSO_CustomerInfo_Med`, select **WebSphere ESB Server v7.5** as the Target runtime environment, and click **Next**, as shown in Figure 2-8 on page 31.

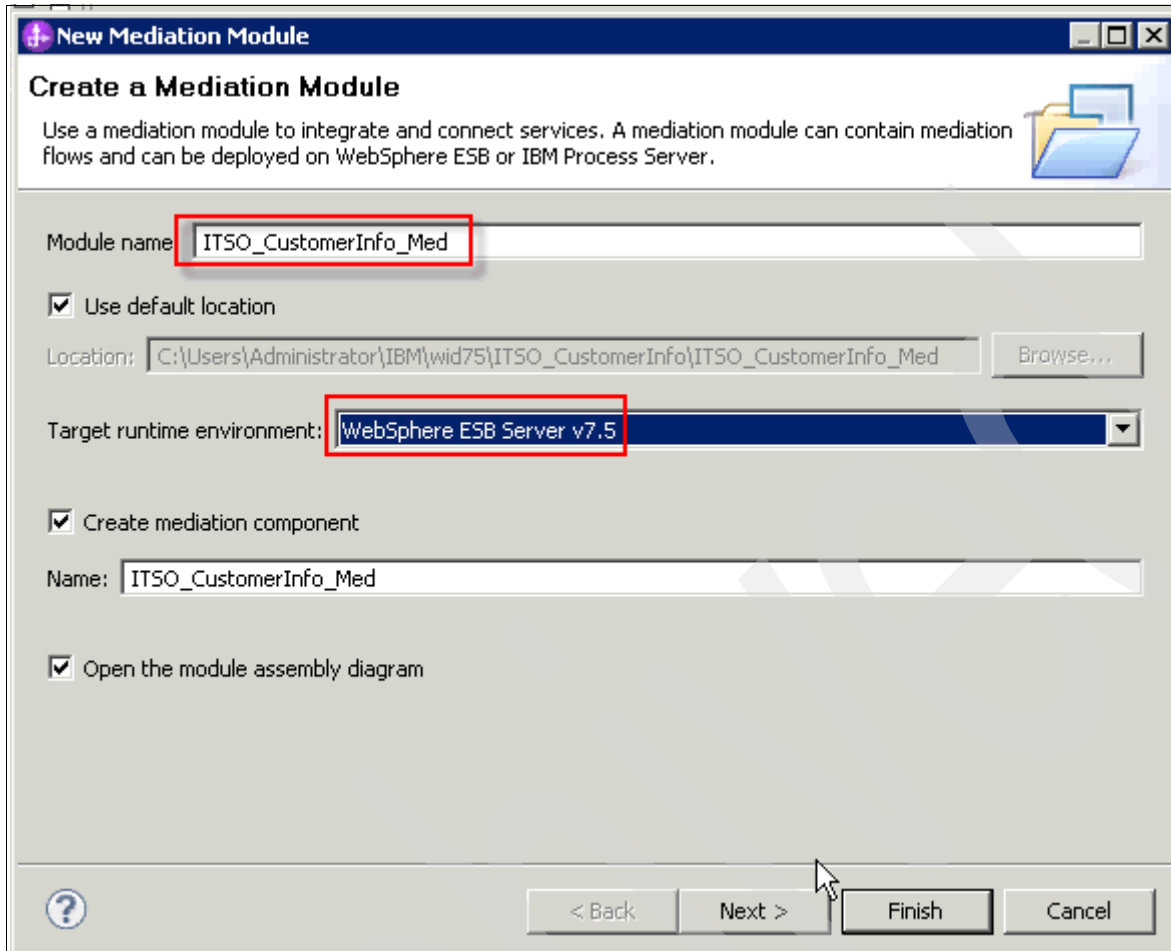


Figure 2-8 Mediation module name and target runtime environment

3. On the next window, select **ITSO_Lib** as a reference library by clicking its check box, as shown in Figure 2-9.

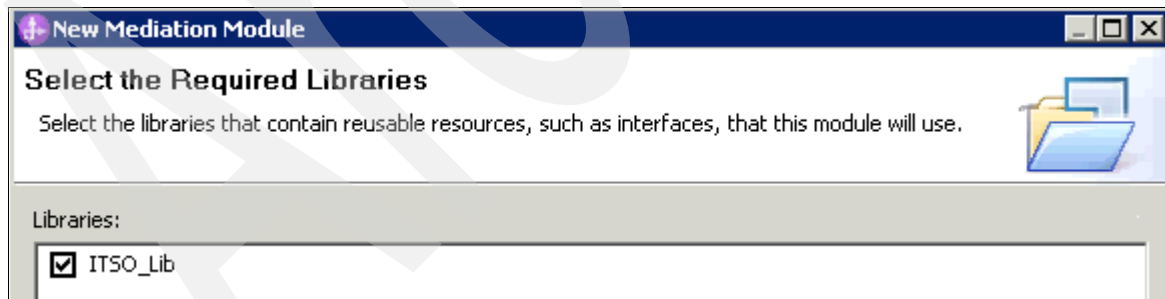


Figure 2-9 Library selection

4. On the next window, select **Lazy parsing** and then click **Finish**. See Figure 2-10 on page 32.

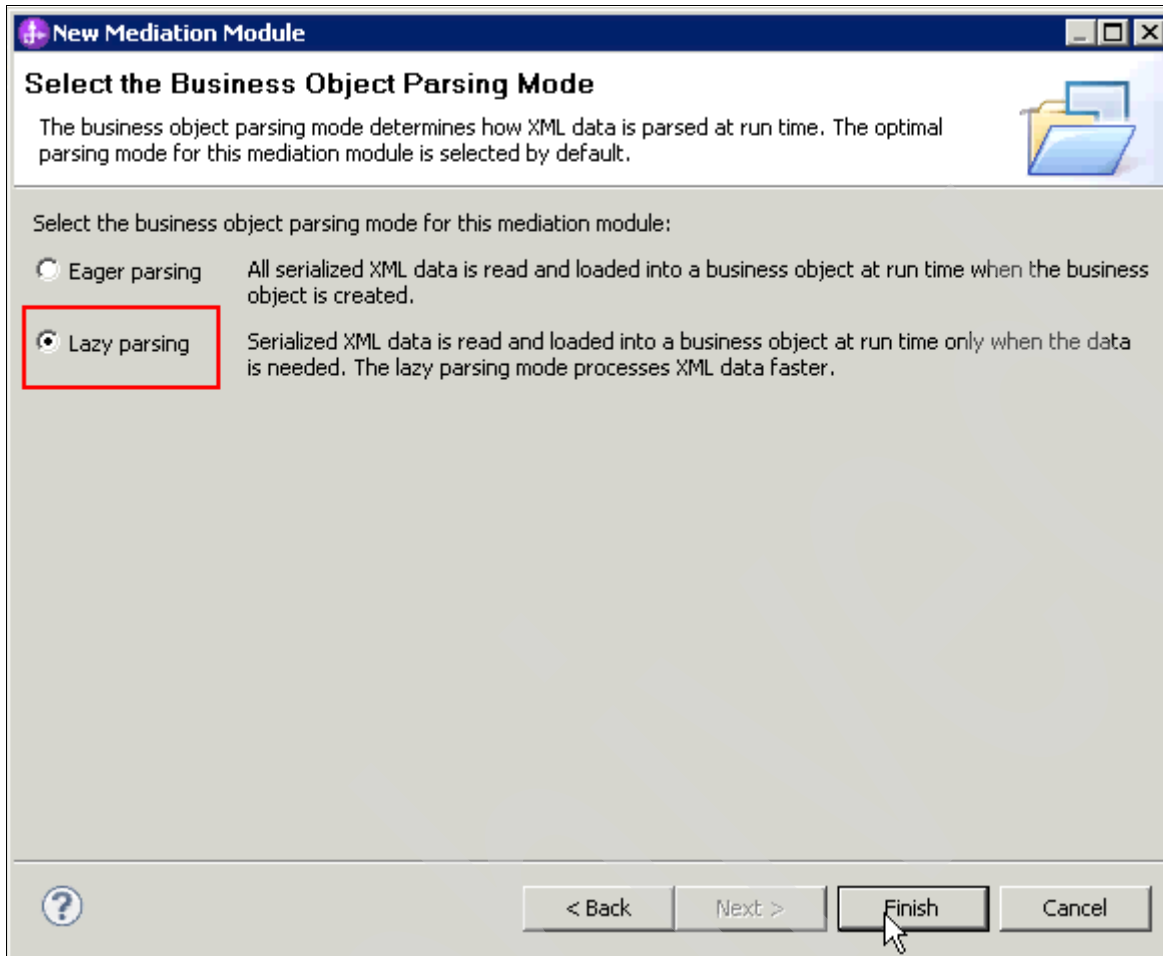


Figure 2-10 Business object parsing mode

- Now, you see the mediation component in the Assembly Diagram, as shown in Figure 2-11.

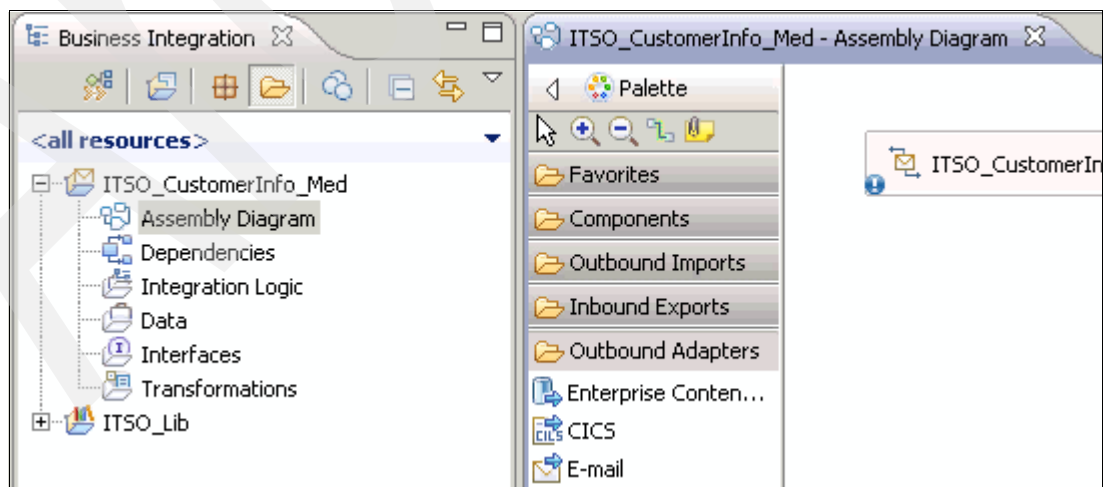


Figure 2-11 Assembly Diagram

2.4.3 Assembling the mediation module

We add the CustomerInfoService Interface in the Assembly Diagram. Our goal is to expose our mediation flow as a SOAP/HTTP service, so that the CSR application and web portal application can access the Web service to add/update/find the customer details. Follow these steps:

1. In the Business Integration view, expand **ITSO_Lib**:
 - a. Select the **CustomerInfoService** interface, drag it onto the Assembly Diagram editor, and release it, as shown as in Figure 2-12.

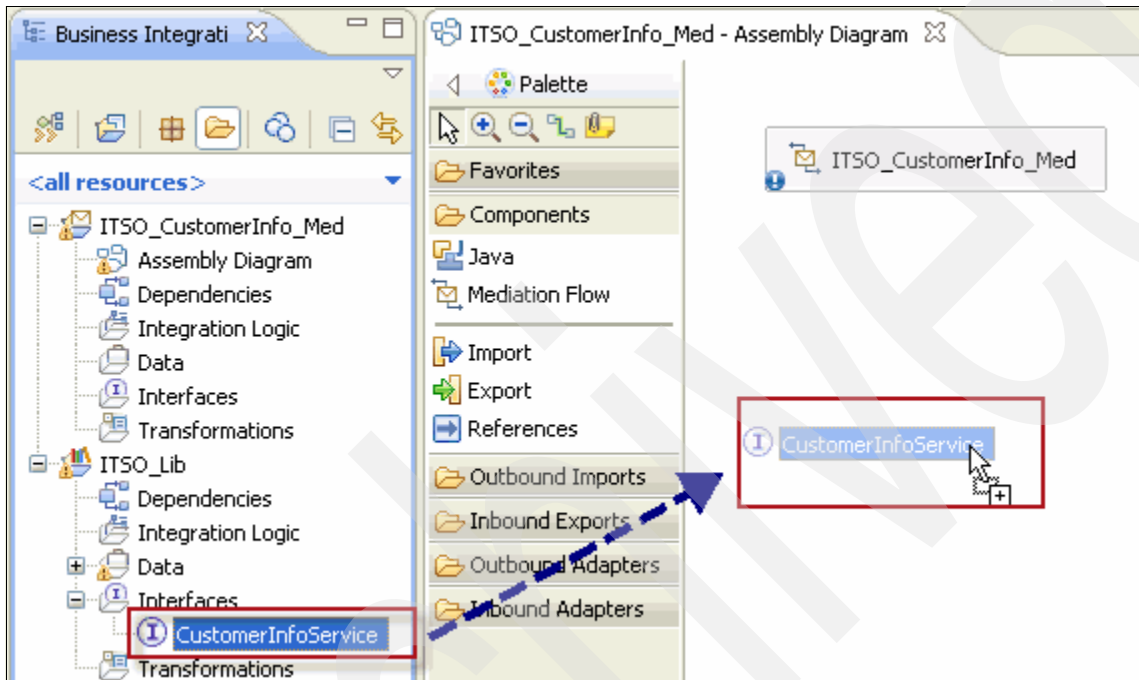


Figure 2-12 Drag the interface to the Assembly Diagram

2. In the Component Creation dialog box, as shown in Figure 2-13, select **Export with Web Service Binding** and click **OK**.

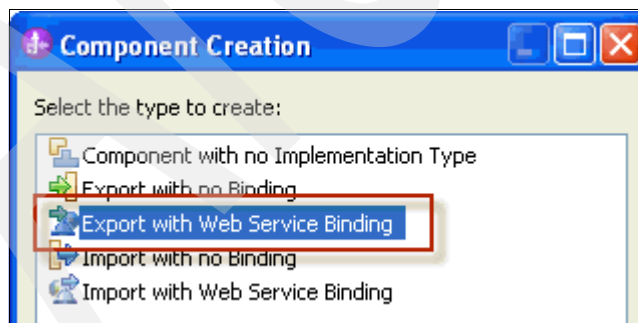


Figure 2-13 Component Creation

3. In the next window, select the transport protocol **SOAP1.1/HTTP** and select **Next**. See Figure 2-14 on page 34.

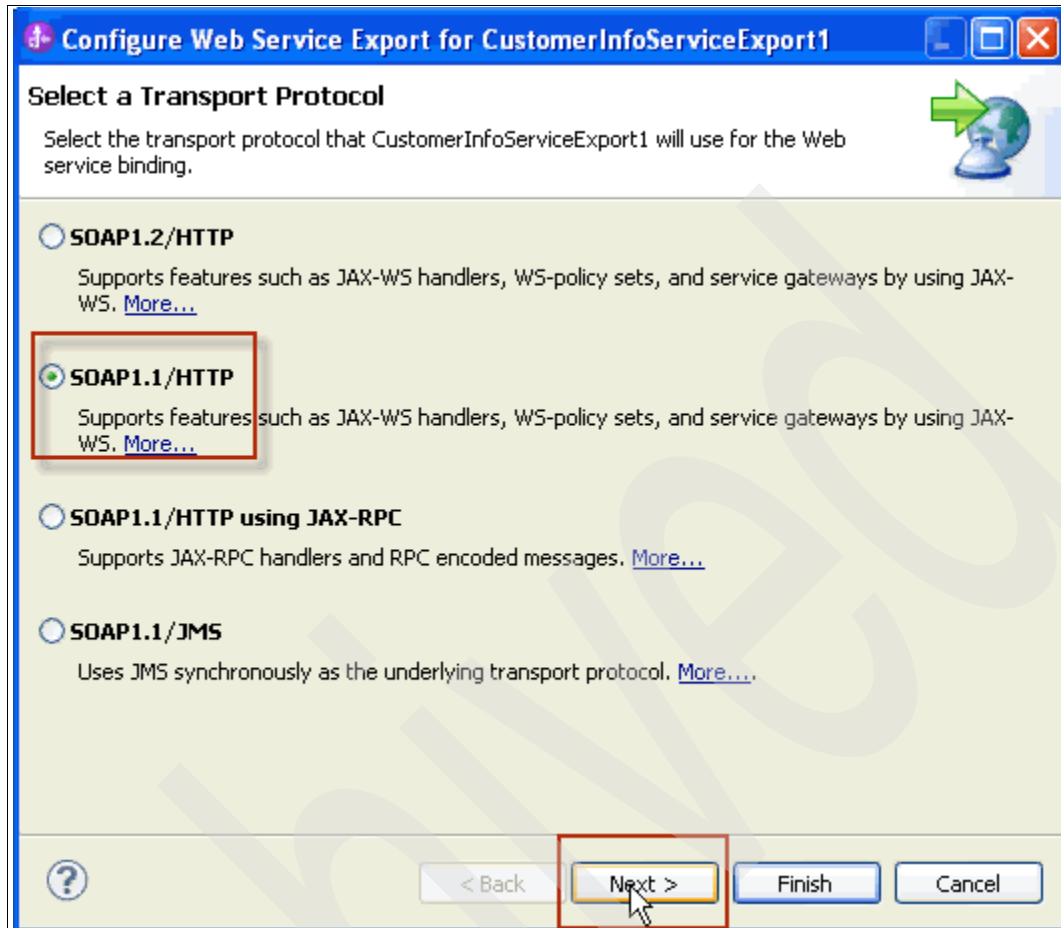


Figure 2-14 Select a Transport Protocol

4. In the next window, select the default value, as shown in Figure 2-15 on page 35.

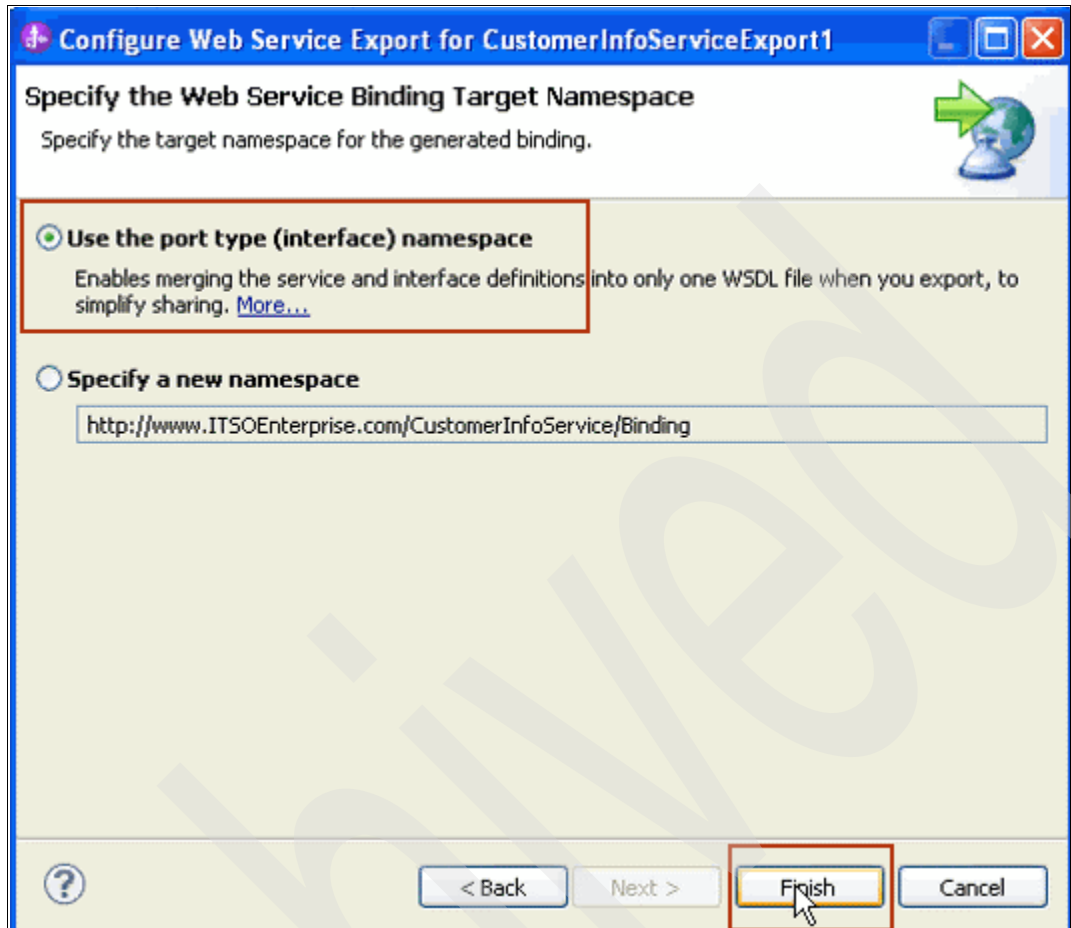


Figure 2-15 Web service binding

5. Click **Finish**.
6. You can see that the Web Services Description Language (WSDL) file is generated in the Web Service Ports of the ITSO_Lib project. In the Assembly Diagram, the CustomerInfoServiceExport1 interface contains a specific symbol for the web service, as shown in Figure 2-16 on page 36.

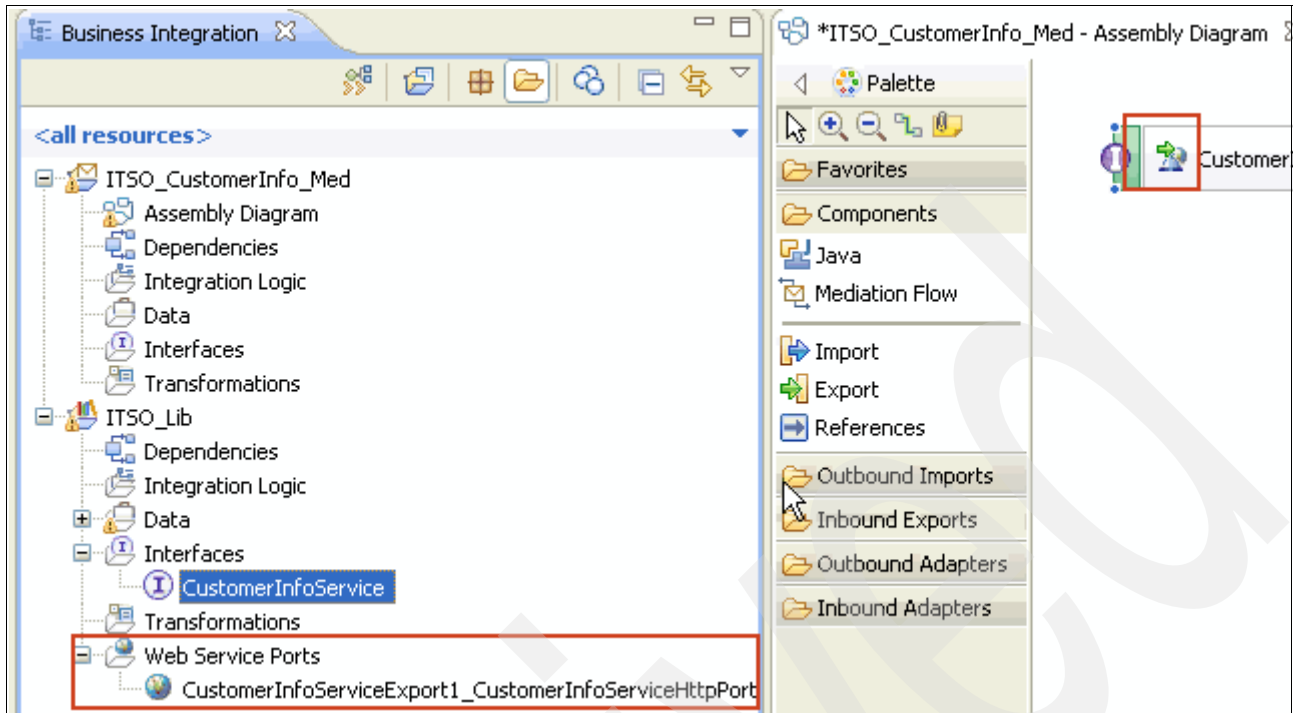


Figure 2-16 Web Service Ports in library and Web service symbol in the Assembly Diagram

7. In the Assembly Diagram, change the name of the Export to CustomerInfoServiceExport.
8. Wire (connect) the export terminal to the mediation component, as shown in Figure 2-17.

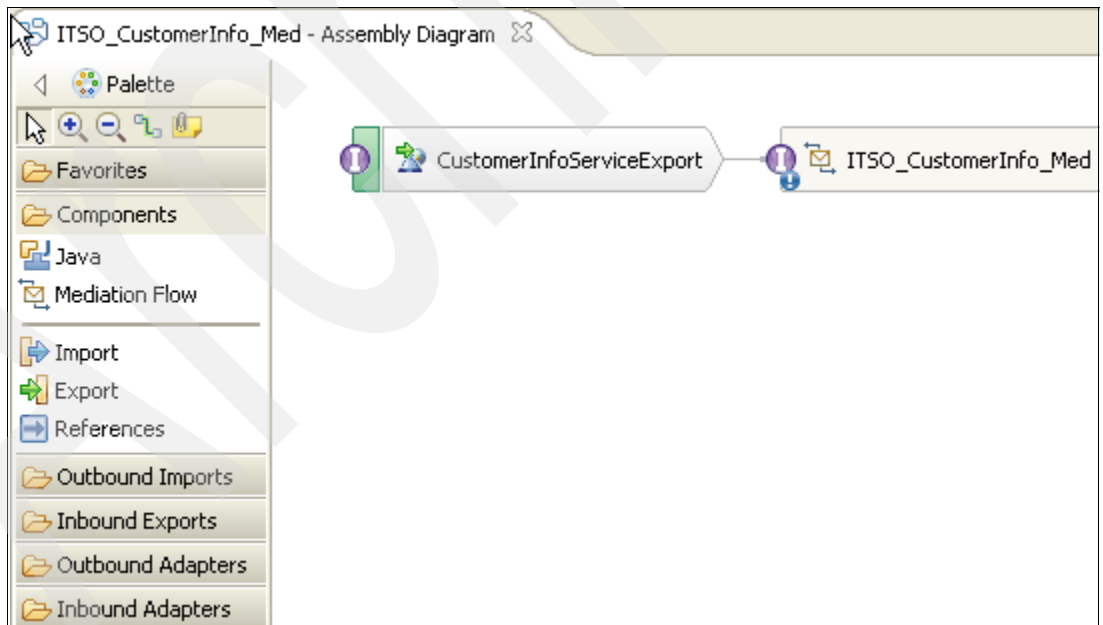


Figure 2-17 Assembly Diagram

2.4.4 Creating a JDBC outbound adapter

We describe how WebSphere Adapter for JDBC 7.5 participates in a global transaction using DB2 Universal JDBC Driver Provider (XA) as a data source for DB2 database.

Before configuring the adapter, you need to complete the following tasks:

1. Create an authentication alias to connect to DB2.
2. Create a data source in the Administration Console.

Creating an authentication alias

The authentication alias needs to be set, because the adapter uses the user name and password that are set in the authentication alias to connect to the database at run time.

Follow these steps to set the authentication alias in the WebSphere ESB Server administration console:

1. In the Integration Designer, go to the **Server** view, right-click the server, and select **Start**.
2. Right-click **WebSphere ESB Server v7.5**, click **Administration** → **Run Administrative Console**, as shown in Figure 2-18.

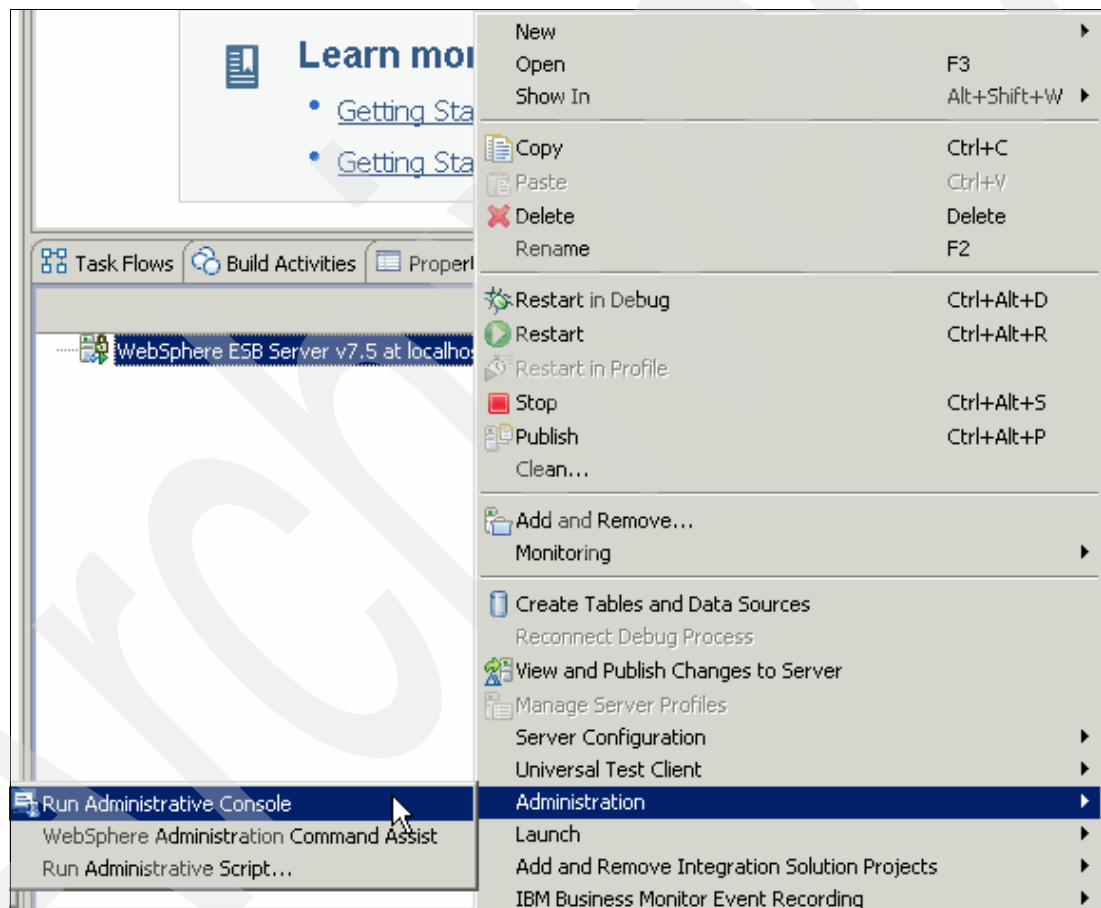


Figure 2-18 Run admin console

3. Log in to the Admin Console by entering the User ID and Password, as shown in Figure 2-20 on page 38.

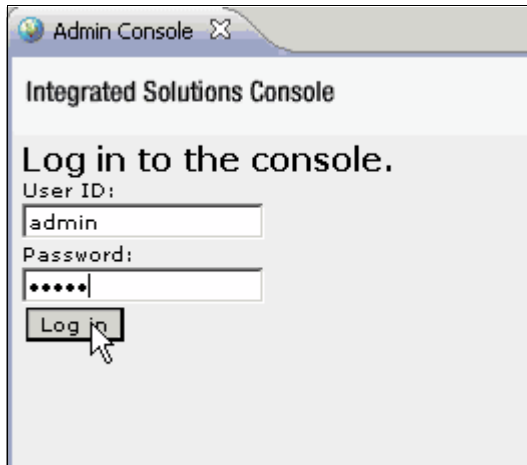


Figure 2-19 Log in to the console

4. On the left side of the console, as shown in Figure 2-20, click **Security** → **Global Security**.

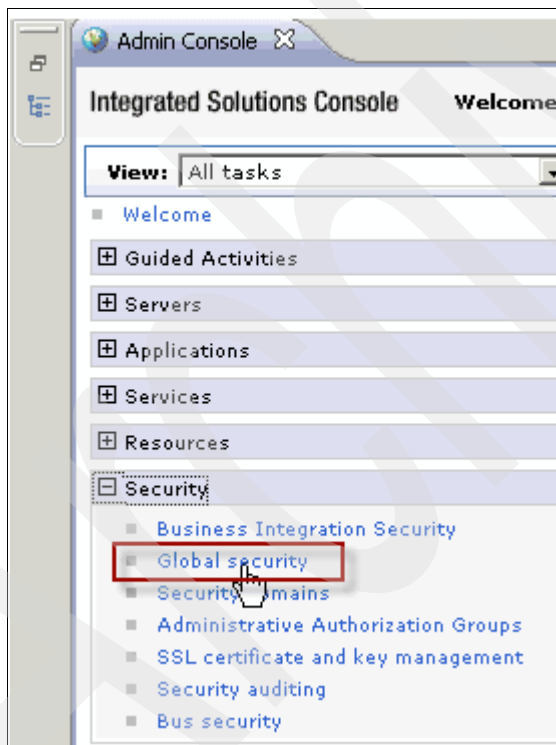


Figure 2-20 Global security

5. On the right side of the console, under the **Java Authentication and Authorization Service**, click **J2C authentication data**. See Figure 2-21 on page 39.

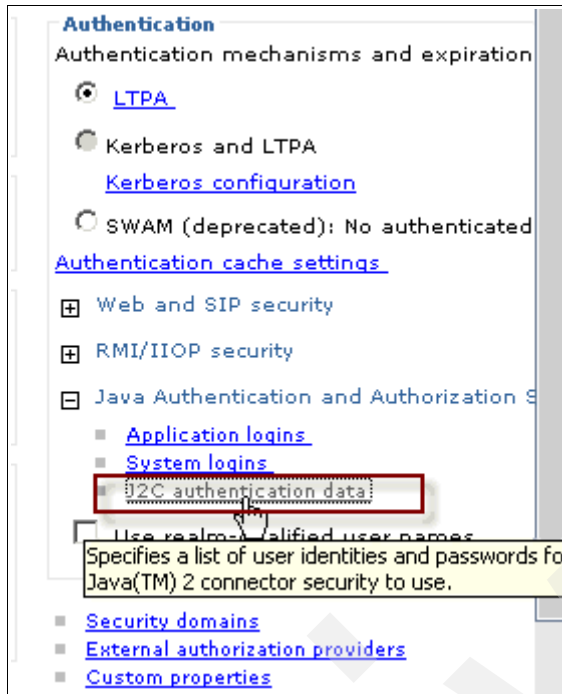


Figure 2-21 J2C authentication data

6. Click **New** to create a new authentication entry, as shown in Figure 2-22.

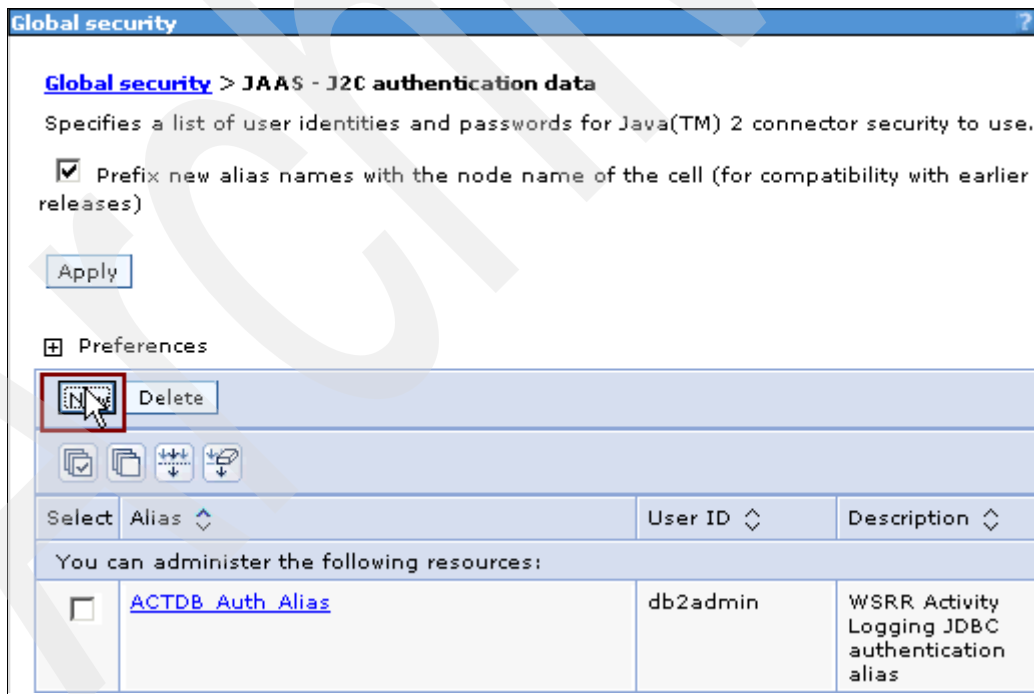


Figure 2-22 New authentication

7. Enter the Alias, User ID, and Password that can connect to the DB2 database (Figure 2-23 on page 40).

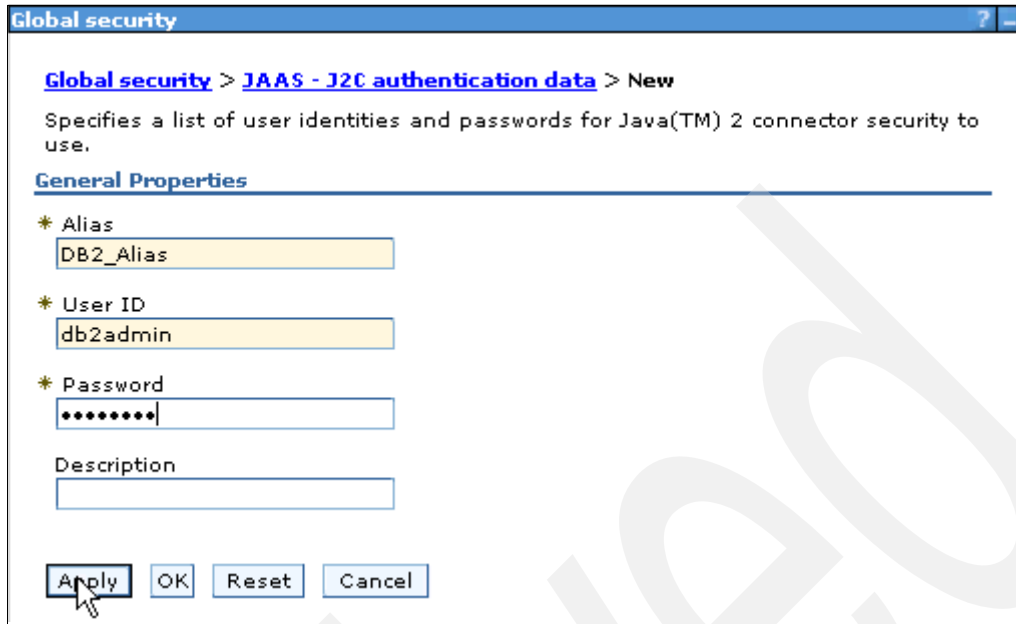


Figure 2-23 Authentication entry

8. Click **Save** to save the changes, as shown in Figure 2-24.

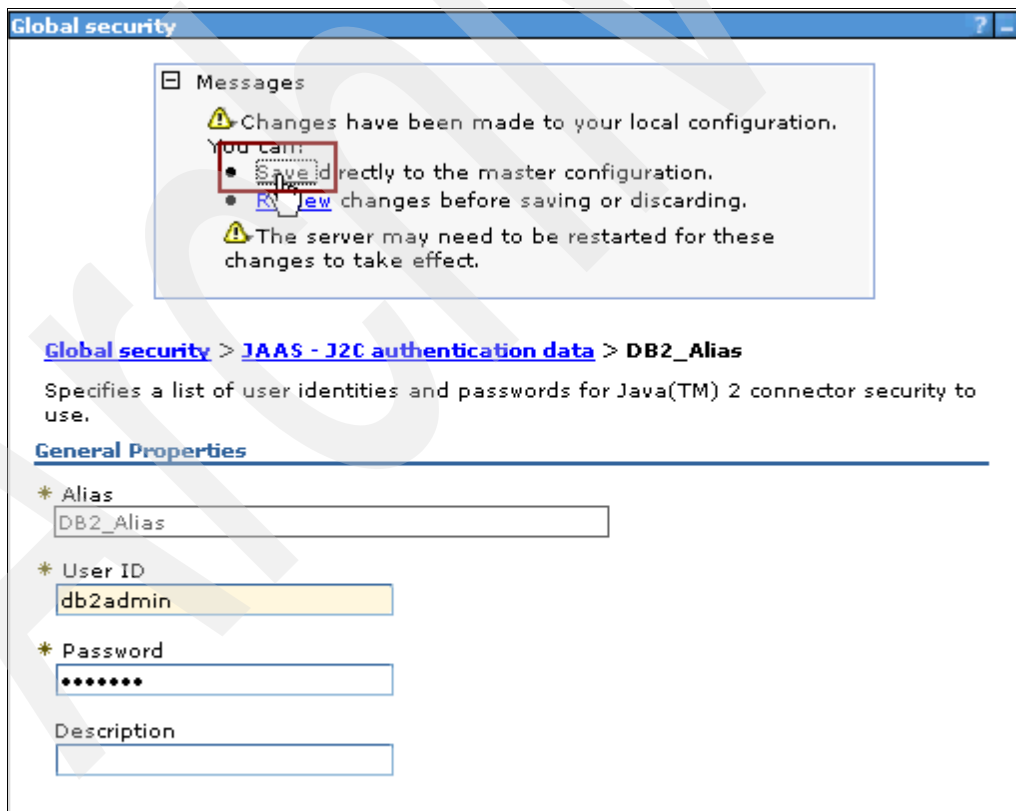


Figure 2-24 Save the configuration

Creating a data source

Follow these steps to create a data source:

1. In the Admin Console, as shown in Figure 2-25, select **Resources** → **JDBC** → **JDBC providers**.

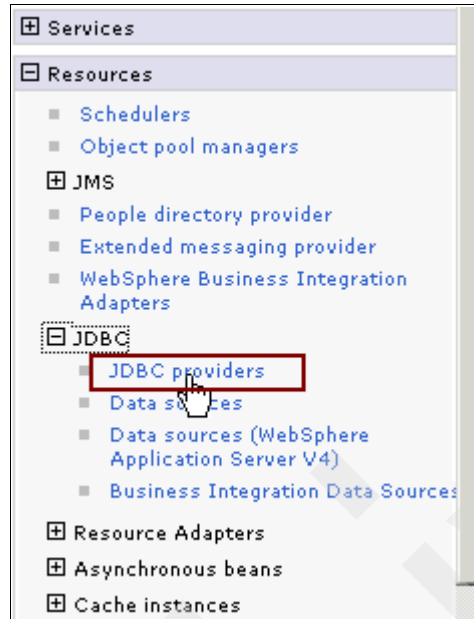


Figure 2-25 JDBC providers

2. Click **DB2 Universal JDBC Driver Provider (XA)**, as shown in Figure 2-26.

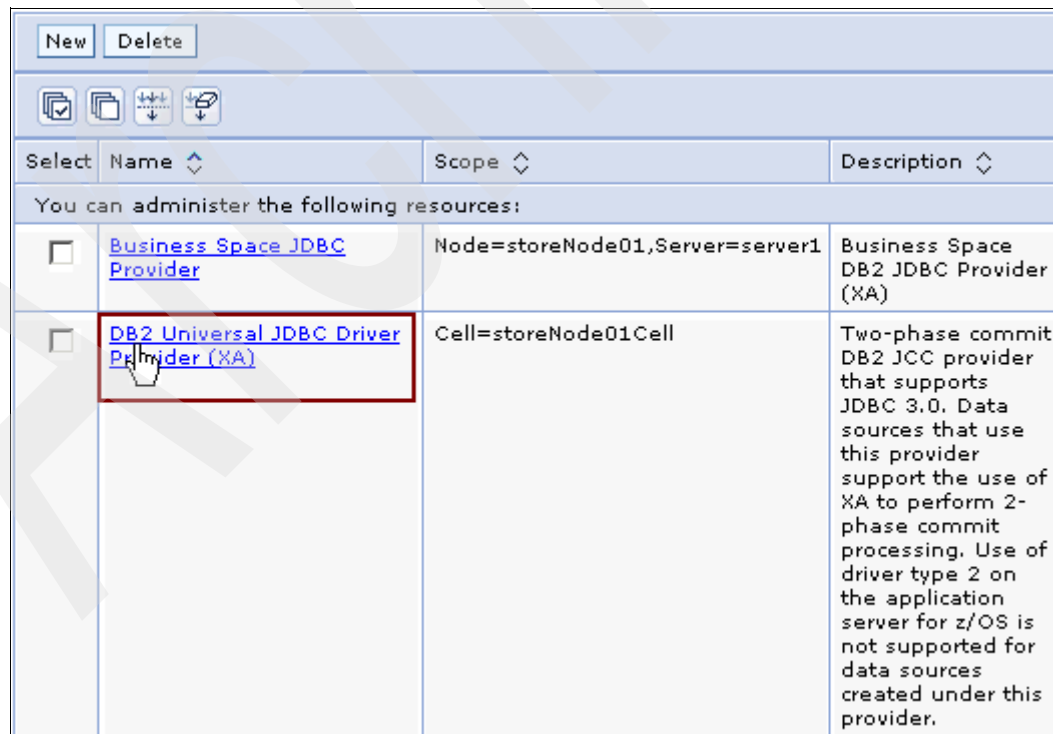


Figure 2-26 Select DB2 Universal JDBC Driver Provider (XA)

3. Under Additional Properties, click **Data sources**, as shown in Figure 2-27.

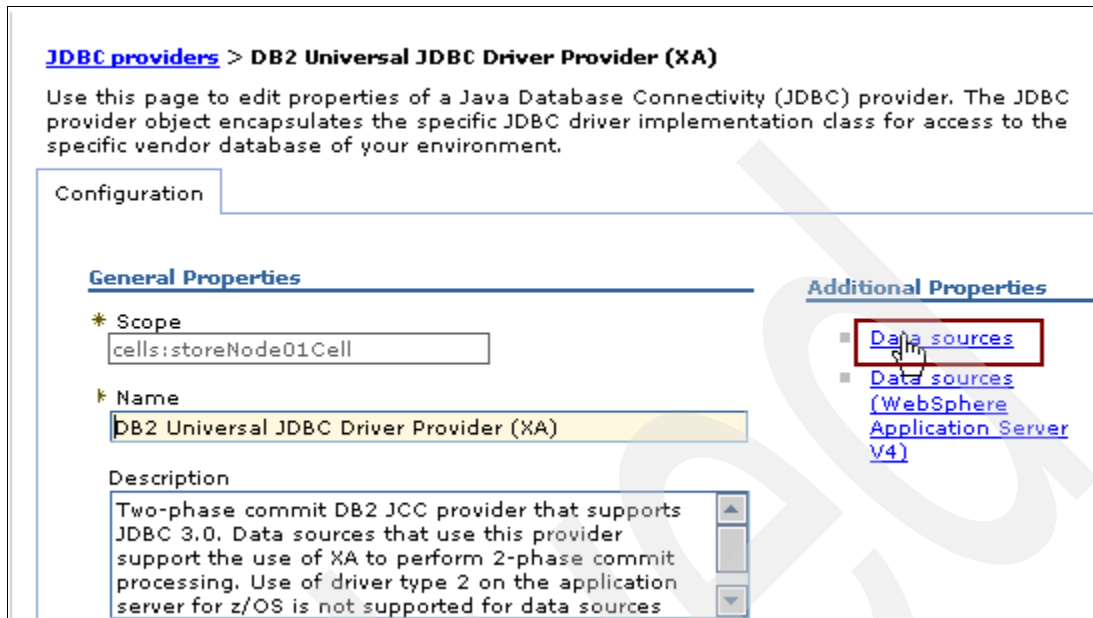


Figure 2-27 Select the data source

4. Click **New**, as shown in Figure 2-28.

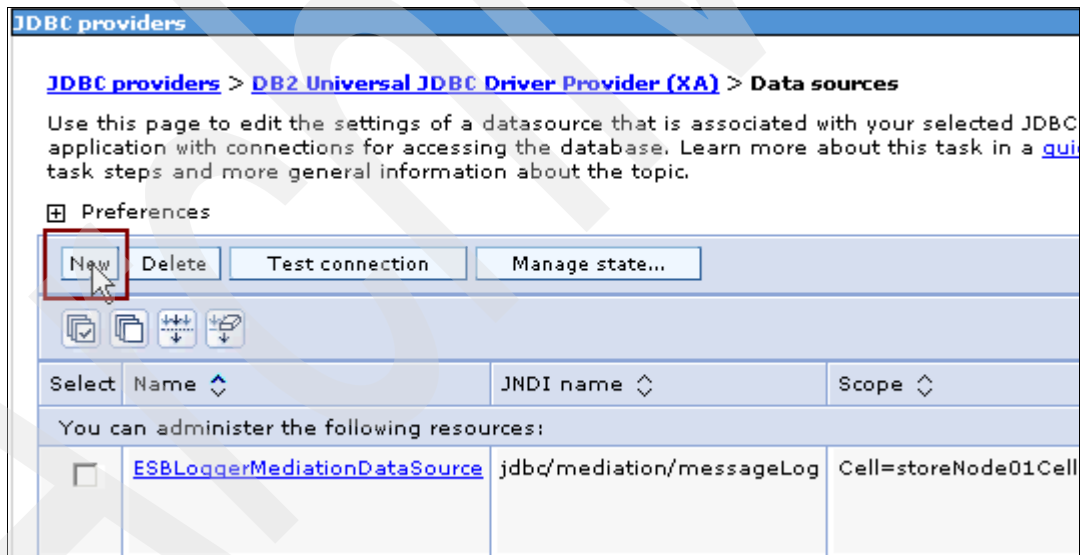


Figure 2-28 Create new data source

5. Enter the JNDI name as `jdbc/CSR` and click **Next** (Figure 2-29 on page 43).

Create a data source

Create a data source

→ **Step 1: Enter basic data source information**

Step 2: Enter database specific properties for the data source

Step 3: Setup security aliases

Step 4: Summary

Enter basic data source information

Set the basic configuration values of a datasource for association with your JDBC provider. A datasource supplies the physical connections between the application server and the database.

Requirement: Use the Datasources (WebSphere(R) Application Server V4) console pages if your applications are based on the Enterprise JavaBeans(TM) (EJB) 1.0 specification or the Java(TM) Servlet 2.2 specification.

Scope
cells:storeNode01Cell

JDBC provider name
DB2 Universal JDBC Driver Provider (XA)

* Data source name
DB2 Universal JDBC Driver XA DataSource

* JNDI name
jdbc/CSR

Next Cancel

Figure 2-29 Enter basic data source information

6. Enter the values for the Database name, Server name, and Port number (Figure 2-30).

Create a data source

Create a data source

Step 1: Enter basic data source information

→ **Step 2: Enter database specific properties for the data source**

Step 3: Setup security aliases

Step 4: Summary

Enter database specific properties for the data source

Set these database-specific properties, which are required by the database vendor JDBC driver to support the connections that are managed through the datasource.

Name	Value
* Driver type	4
* Database name	ITSO_CSR
* Server name	localhost
* Port number	50000

Use this data source in container managed persistence (CMP)

Previous Next Cancel

Figure 2-30 Database specific properties for the data source

- As described in Figure 2-31, under the Component-managed authentication alias, select the name of the authentication alias that we previously created from the drop-down list and click **Next**.

Create a data source

Step 1: Enter basic data source information

Step 2: Enter database specific properties for the data source

→ **Step 3: Setup security aliases**

Step 4: Summary

Setup security aliases

Select the authentication values for this resource.

Authentication alias for XA recovery
(none)

Component-managed authentication alias
storeNode01/DB2_Alias

Mapping-configuration alias
(none)

Container-managed authentication alias
(none)

Note: You can create a new J2C authentication alias by accessing one of the following links. Clicking on a link will cancel the wizard and your current wizard selections will be lost.

[Global J2C authentication alias](#)
[Security domains](#)

Previous Next Cancel

Figure 2-31 Security alias

- Click **Finish**, as shown in Figure 2-32.

Database name	ITSO_CSR
Server name	localhost
Port number	50000
Use this data source in container managed persistence (CMP)	true
Authentication alias for XA recovery	(none)
Component-managed authentication alias	storeNode01/DB2_Alias
Mapping-configuration alias	(none)
Container-managed authentication alias	(none)

Previous **Finish** Cancel

Figure 2-32 Review all properties

9. Click **Save** to save the changes to the master configuration (Figure 2-33).

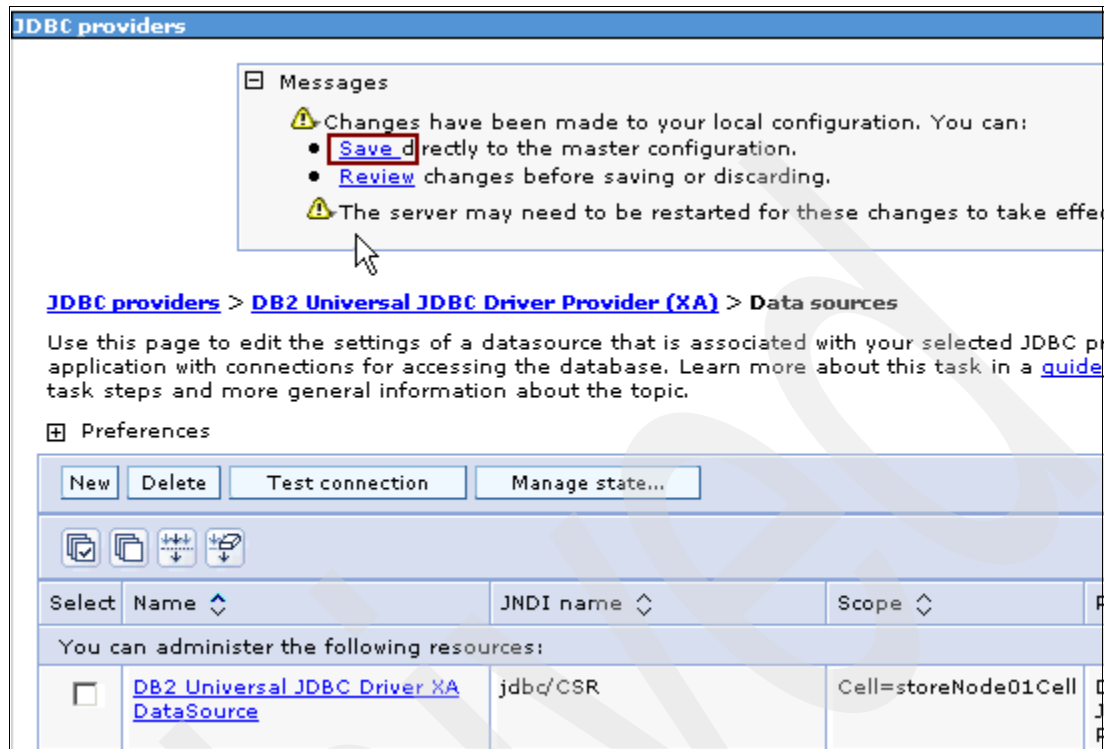


Figure 2-33 Save changes to the master configuration

10. As shown in Figure 2-34, select the check box next to the data source that we have just created. Click **Test connection**.

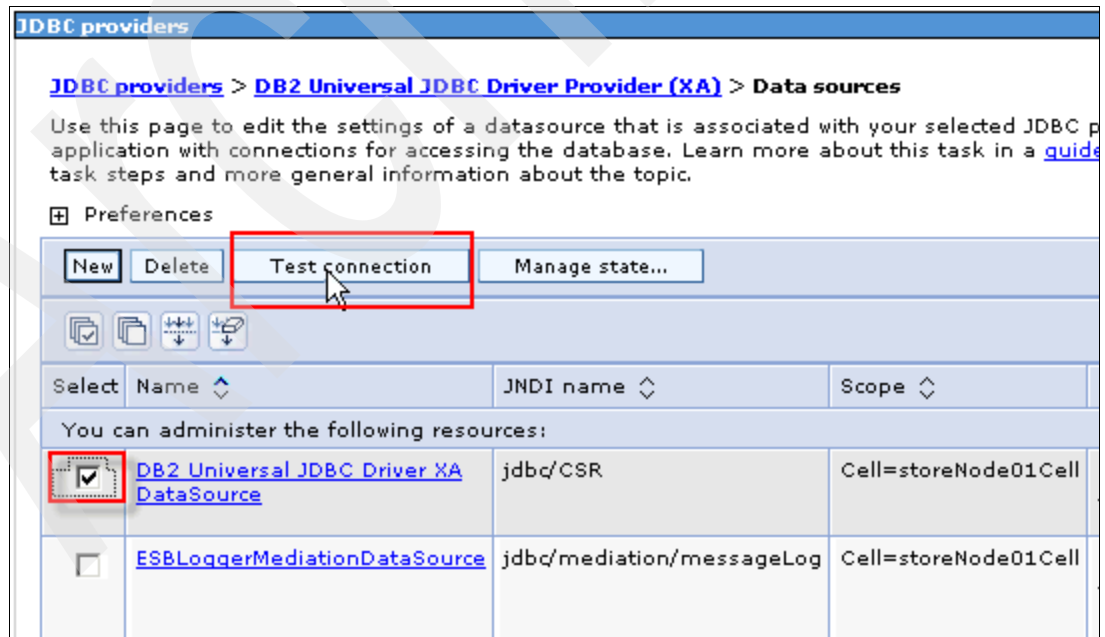


Figure 2-34 Test the connection for the data source

11. The connection succeeds, as indicated by the message that is shown in Figure 2-35 on page 46.

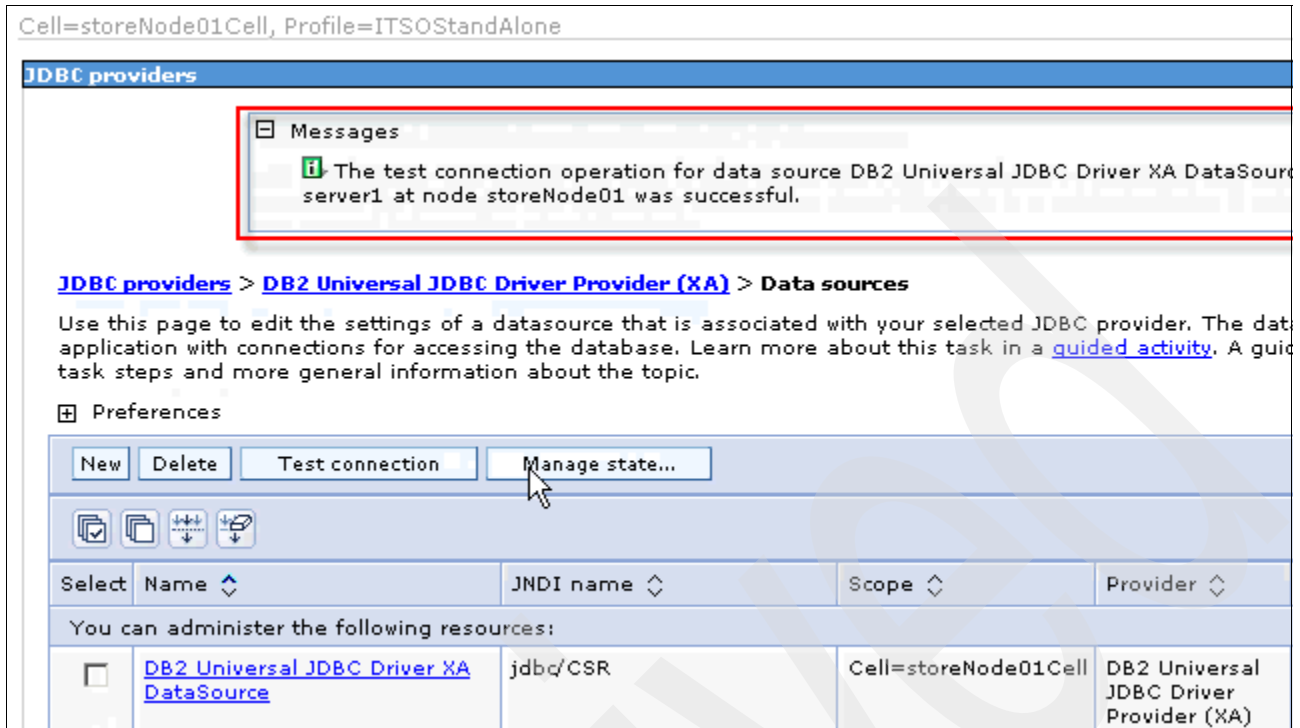


Figure 2-35 Success message

Other databases: Follow steps 1 through 11 two more times to configure the data sources for the two other databases and provide the following Java Naming and Directory Interface (JNDI) names:

- ▶ jdbc/WWW
- ▶ jdbc/MST

After creating the two other data sources for the WWW and MST DB2 databases, you can see the three data source configurations, as shown in Figure 2-36.

Select	Name	JNDI name	Scope	Provider
You can administer the following resources:				
<input type="checkbox"/>	DB2 Universal JDBC Driver XA DataSource	jdbc/CSR	Cell=storeNode01Cell	DB2 Universal JDBC Driver Provider (XA)
<input type="checkbox"/>	DB2 Universal JDBC Driver XA DataSource	jdbc/MST	Cell=storeNode01Cell	DB2 Universal JDBC Driver Provider (XA)
<input type="checkbox"/>	DB2 Universal JDBC Driver XA DataSource	jdbc/WWW	Cell=storeNode01Cell	DB2 Universal JDBC Driver Provider (XA)

Figure 2-36 Three data source configurations for three separate databases

Configuring the JDBC outbound adapter in the mediation module

We demonstrate how to configure the JDBC outbound adapter from the Assembly Diagram in the mediation module:

1. Switch to the Business Integration perspective in Integration Designer.

2. In the Assembly Diagram, click **Outbound Adapters** → **JDBC** and drag it to the Assembly Diagram, as shown in Figure 2-37.

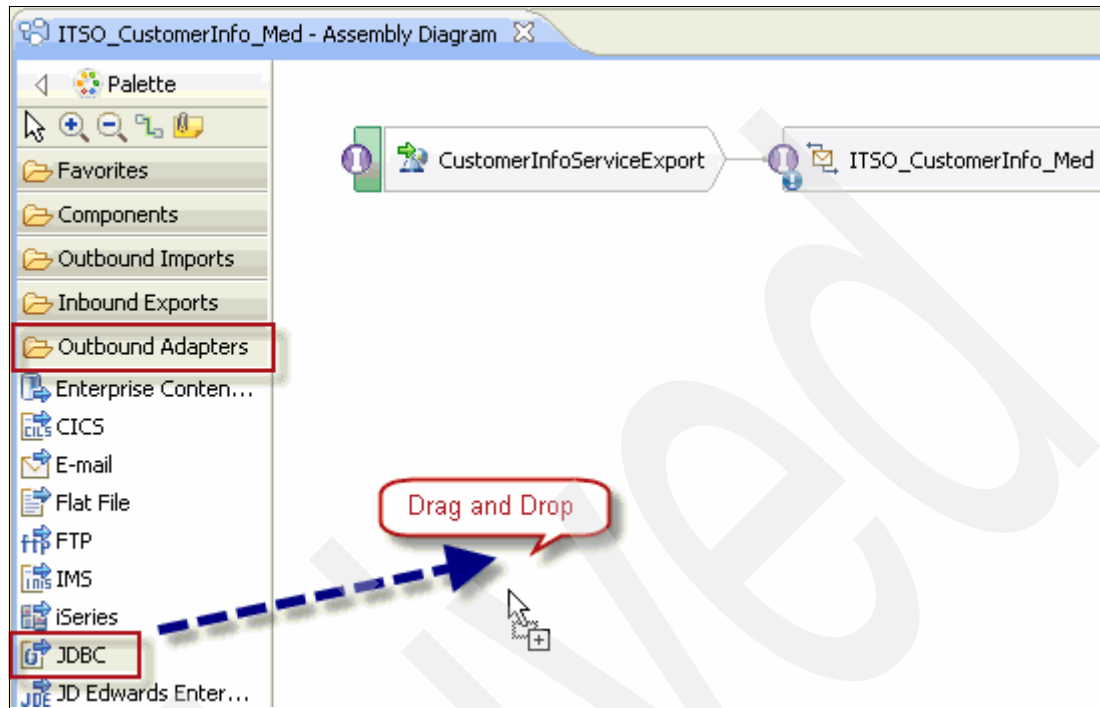


Figure 2-37 JDBC outbound adapter

3. Select **IBM WebSphere Adapter for JDBC** and click **Next**, as shown in Figure 2-38.

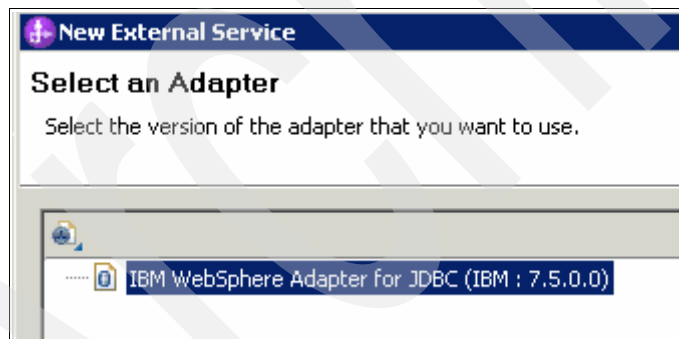


Figure 2-38 Select JDBC adapter

4. For the Target runtime environment, select **WebSphere ESB Server v7.5** and click **Next** (Figure 2-39 on page 48).

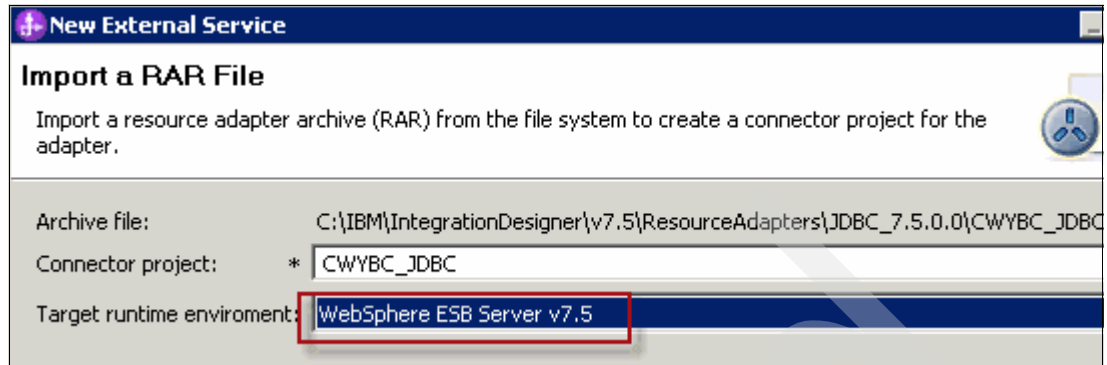


Figure 2-39 Select the target runtime environment

5. Add the JDBC driver JAR file locations for `db2jcc.jar` and `db2jcc_license_cu.jar` from the local system, as shown in Figure 2-40.

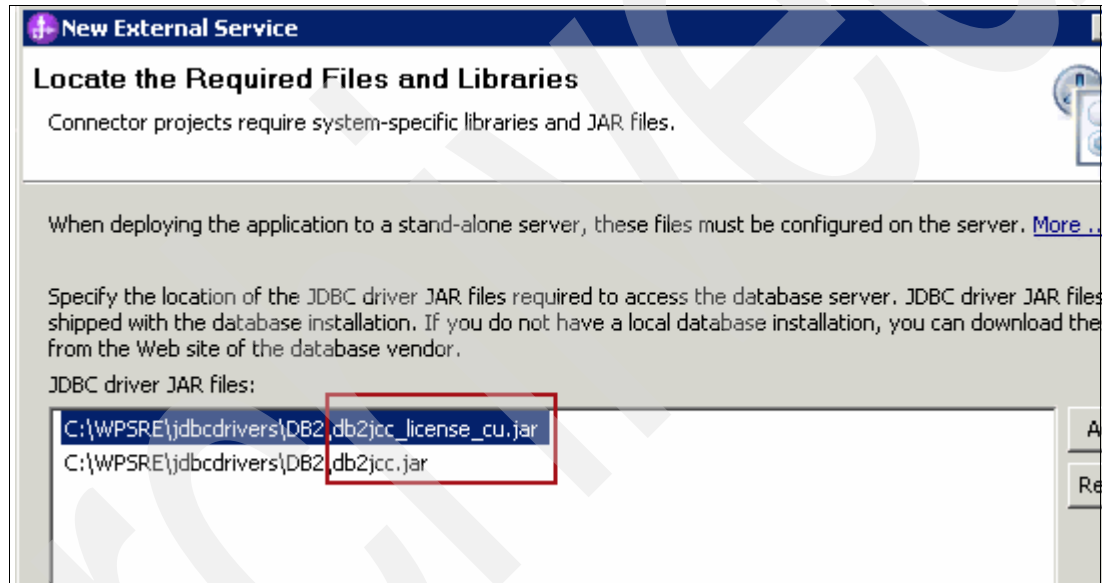


Figure 2-40 Locate JAR files

6. In the next window, select DB2 UDB with the appropriate version of database and enter values in the Database, Host name, Port number, User name, and password fields and click **Next**. See Figure 2-41 on page 49.

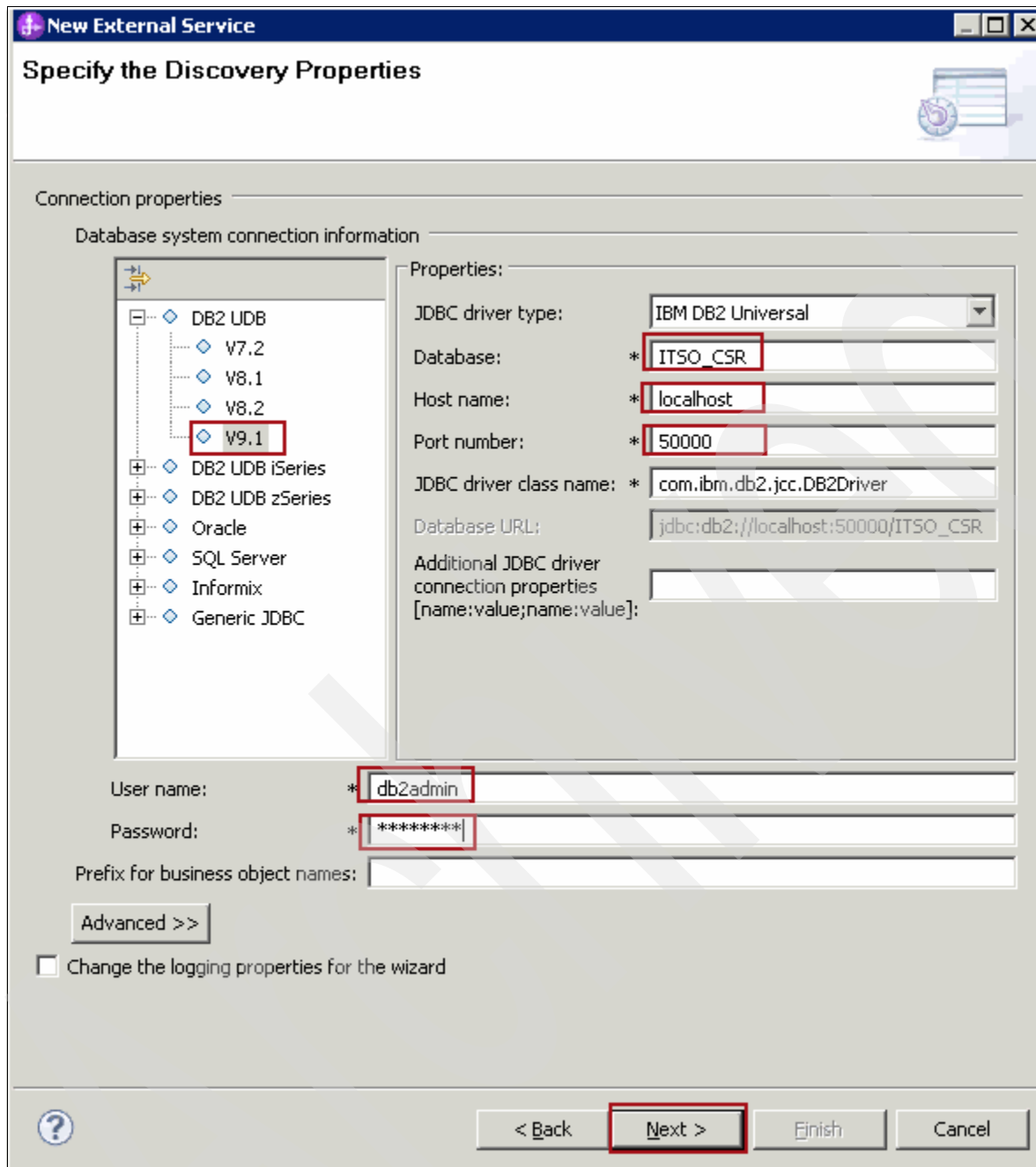


Figure 2-41 Database connection properties

7. In the next window, click **Run Query**, which will list the tables, stored procedure, views, and synonyms for each schema in the database (Figure 2-42 on page 50).

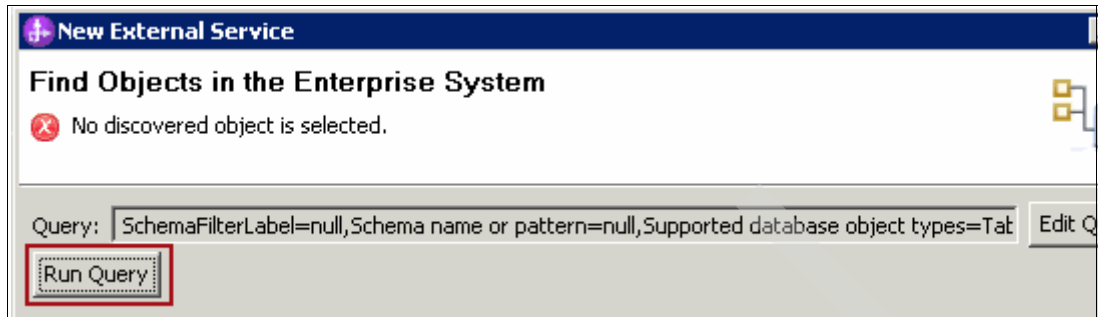


Figure 2-42 Find the objects

8. Expand **CSR** → **Tables** → **CUSTOMERS** and click > to add CUSTOMERS to the Selected objects list and then click **Next**. See Figure 2-43.

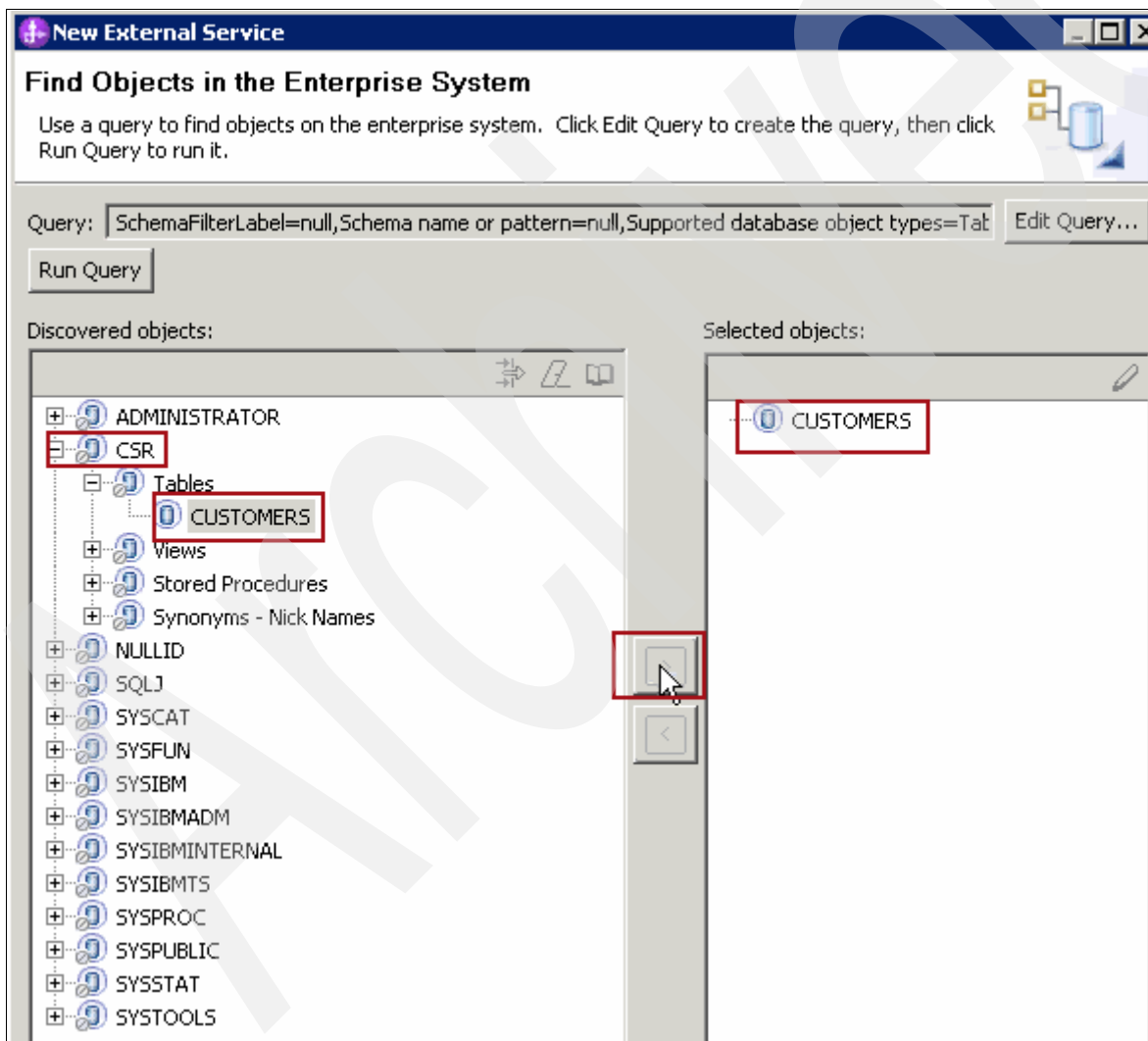


Figure 2-43 Select database table

9. In the next window, select the operation that you want to use and click **Next**. In this scenario, we use two operations: Create customers and Update customers (Figure 2-44 on page 51).

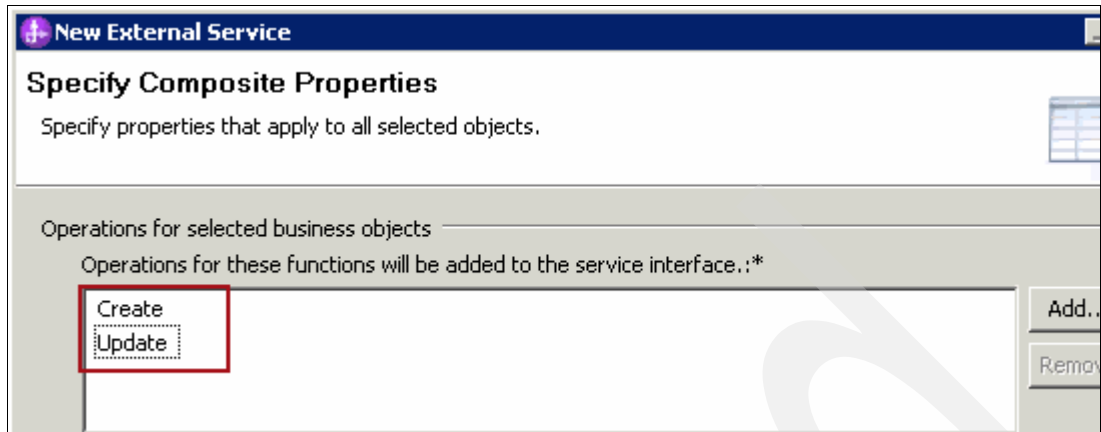


Figure 2-44 Select operations to add to the service interface

10. In the next window, select the security credential as **Other**, select **Specify predefined XA DataSource** and for the name of the XA DataSource JNDI name, enter `jdbc/CSR` and click **Next**, as shown in Figure 2-45.

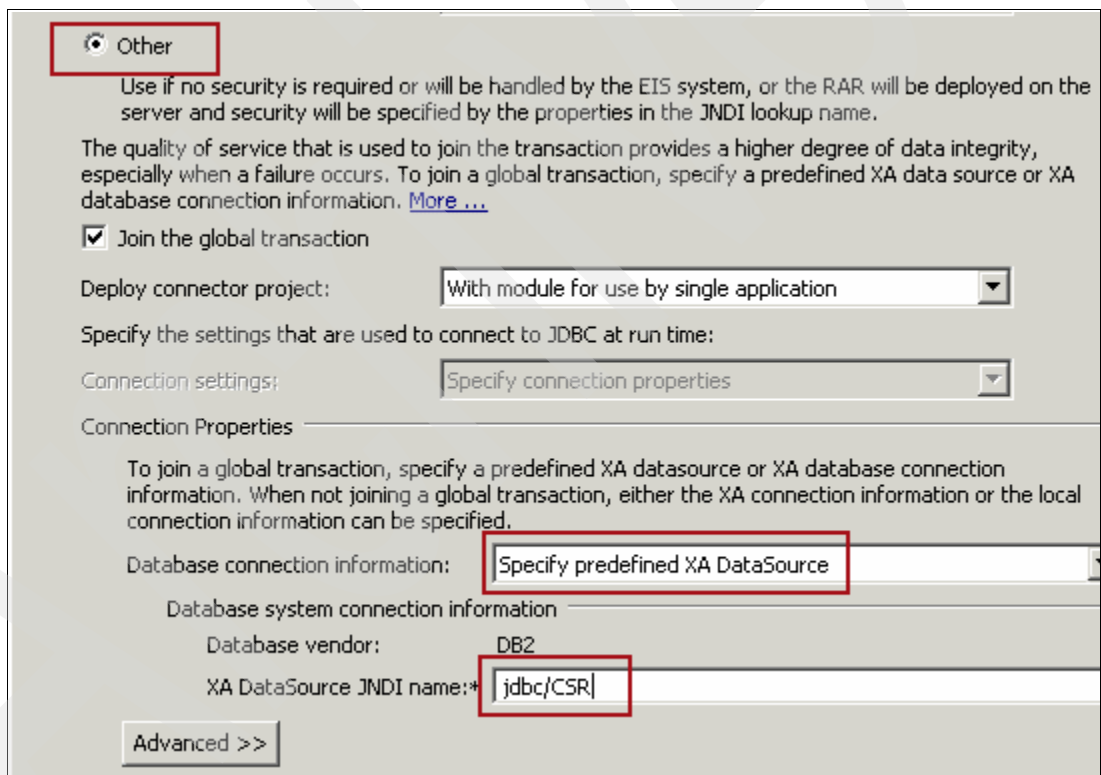


Figure 2-45 Specify predefined XA data source

11. In the next window, for the Name, enter `CSR_JDBCImport` and click **Save business objects to a library**. For Library, enter `ITSO_Lib` and click **Finish** (Figure 2-46 on page 52).

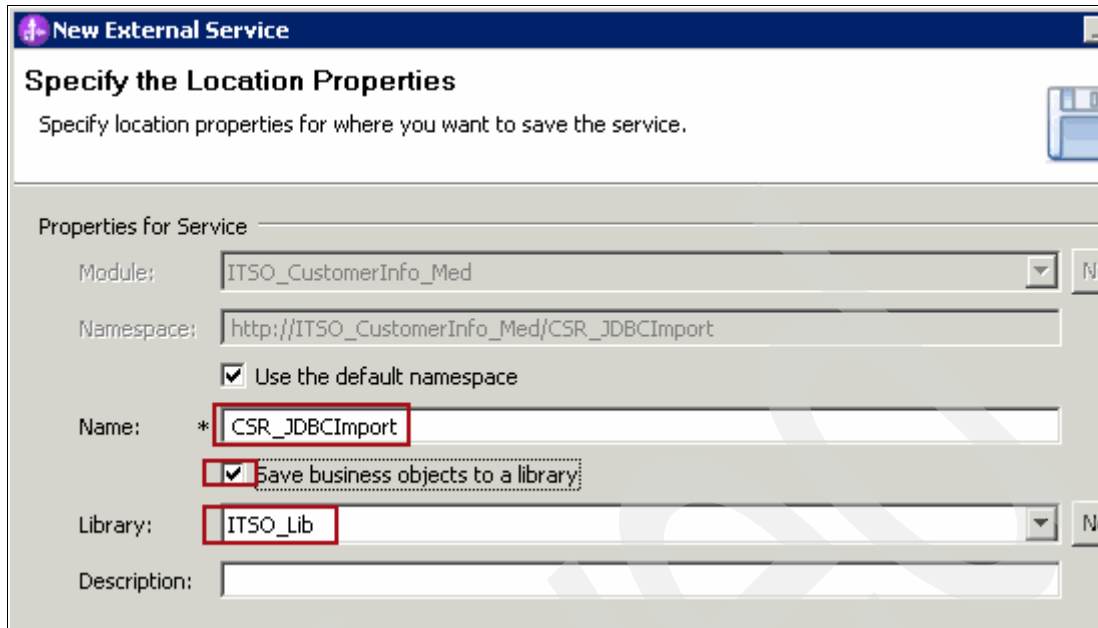


Figure 2-46 Specify the Location Properties

12. As a result, as shown in Figure 2-47, you see the CSR_JDBCImport interface that has been created as a reference in the Assembly Diagram. Also, the generated business objects are placed in the ITSO_Lib library.

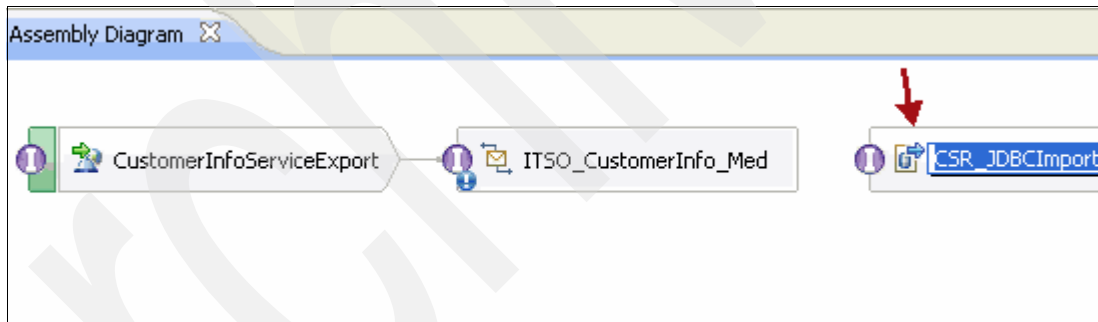


Figure 2-47 Assembly Diagram

Next steps: Follow steps 1 through 12 to configure the JDBC outbound adapters for the two other DB2 databases: ITSO_WWW and ITSO_MST.

Name the interfaces:

- ▶ WWW_JDBCImport
- ▶ MST_JDBCImport

After creating three JDBC outbound adapters and wiring reference interfaces to the mediation flow, your Assembly Diagram looks similar to Figure 2-48 on page 53.



Figure 2-48 Assembly Diagram with end-to-end connectivity

2.4.5 Implementing the mediation module

We implement the mediation module in which we develop the logic to add and update customer details in three DB2 databases: CSR, WWW, and MST. To find customer details, we use the master (MST) DB2 database to fetch all records for a customer.

We follow these steps to implement the mediation module.

Adding customer details: Mediation flow development

Perform the following steps to add customer details in the mediation flow development:

1. In the Assembly Diagram, double-click **ITSO_CustomerInfo_Med**. A pop-up window opens. Click **Yes**.
2. In the next window, you see the interface and references with operations, as shown in Figure 2-49.

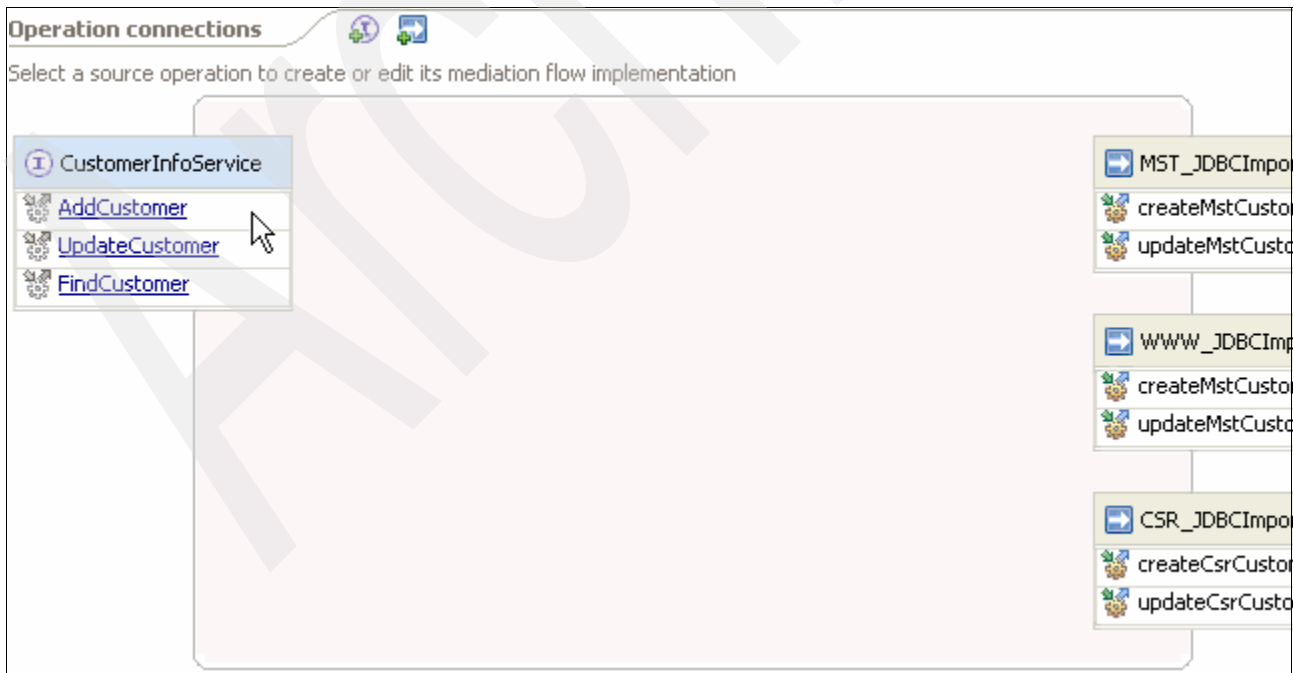


Figure 2-49 Operation connections window

- Right-click **AddCustomer** and select **Open Implementation**, as shown in Figure 2-50 on page 54.

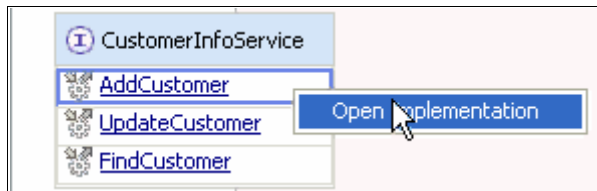


Figure 2-50 Add customer

- Select **Service Integration**, as described in Figure 2-51.

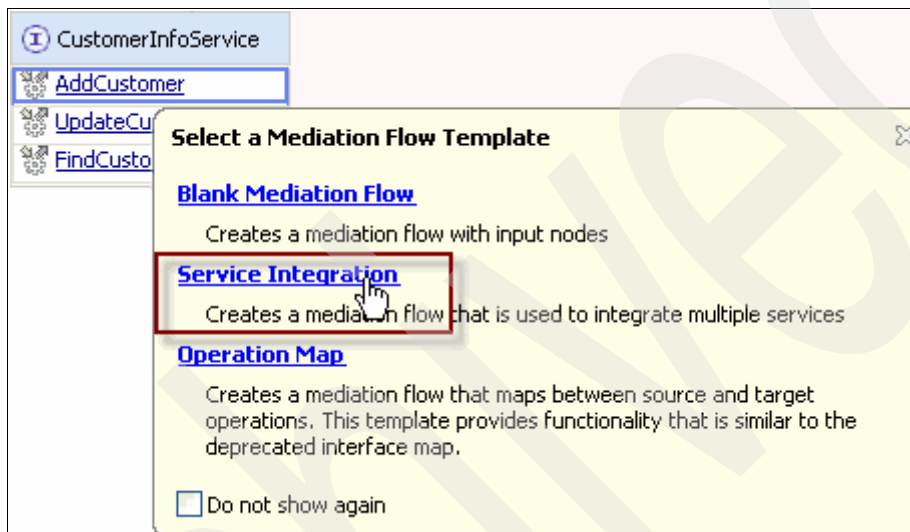


Figure 2-51 Service Integration

- In the next window, add **Reference** and **Target Operation**, starting with create, as shown in Figure 2-52, and click **OK**.

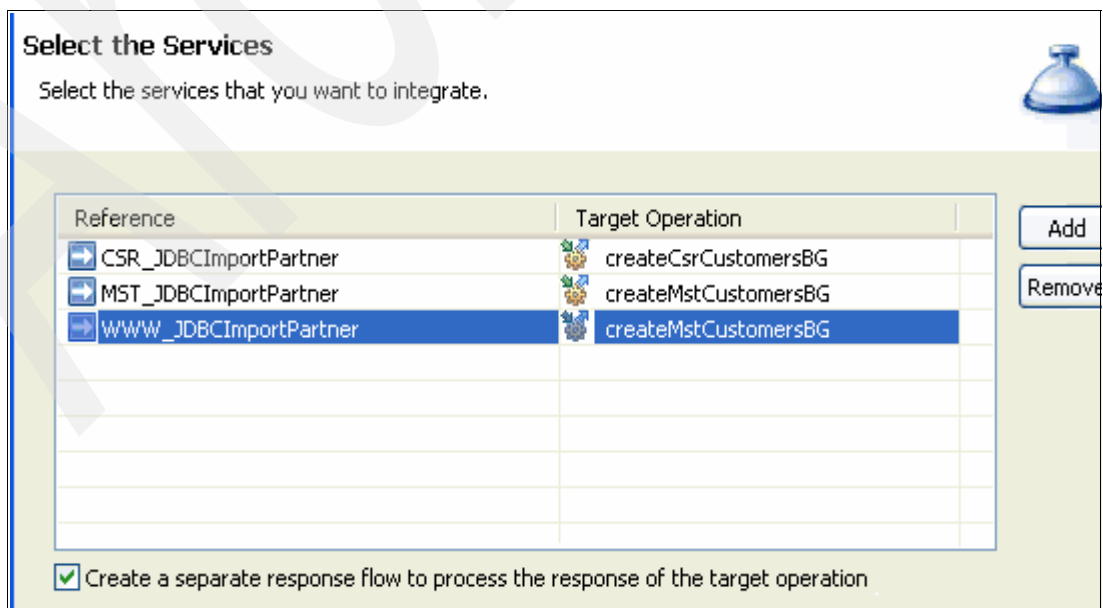


Figure 2-52 Select the services

- In the next window, add the mediation primitives to the mediation flow, as shown in Figure 2-53.

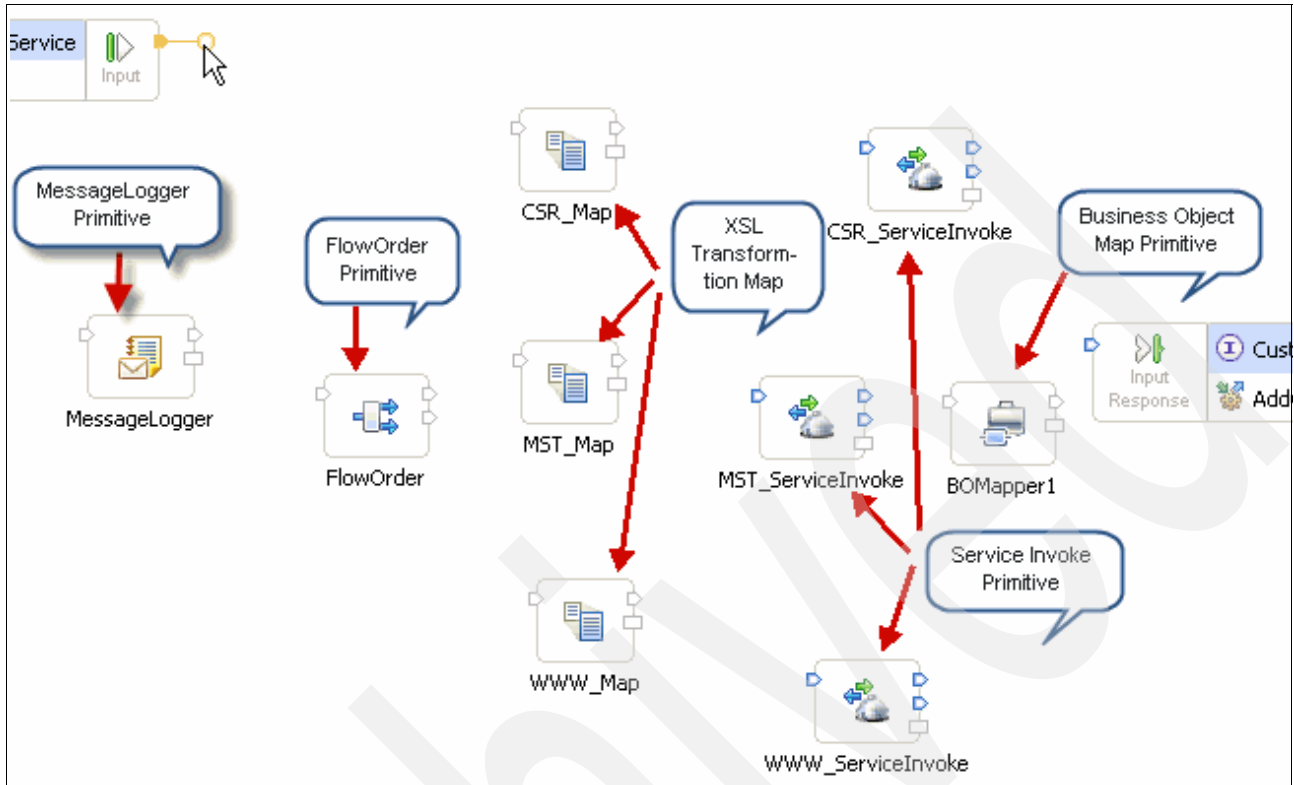


Figure 2-53 Mediation flow

- After wiring each mediation primitive, the flow looks similar to the flow that is shown in Figure 2-54 on page 56.

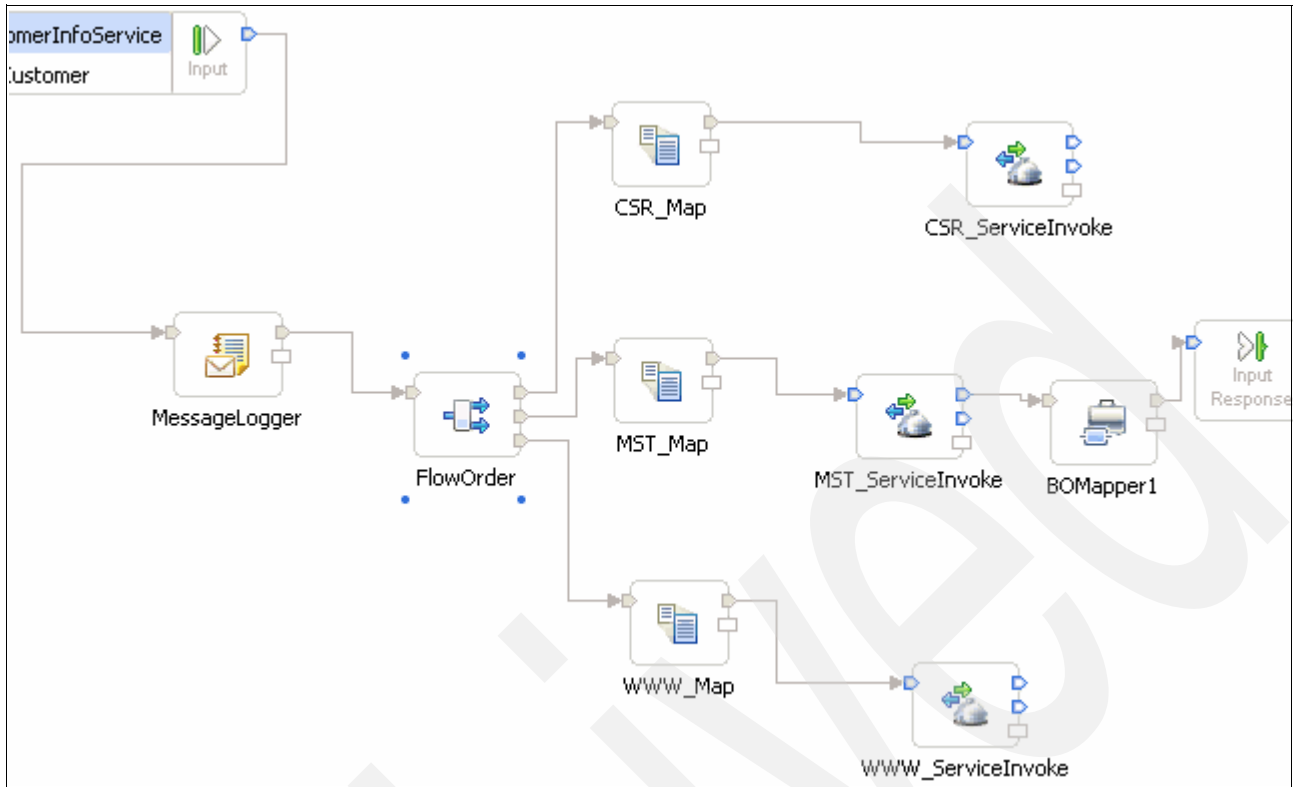


Figure 2-54 Wired mediation primitives in flow

8. Add the **Fail** primitive to throw an explicit error from the mediation flow and add the **Stop** primitive. Stop the flow after getting a success response from CSR and WWW DB2 databases. See Figure 2-55.

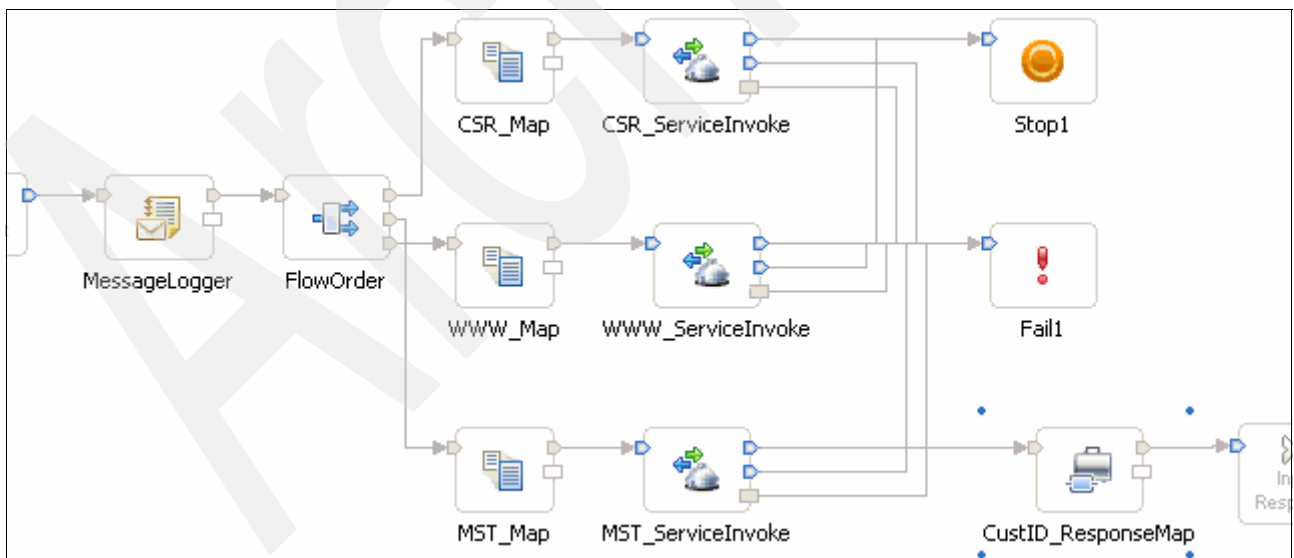


Figure 2-55 Mediation flow

9. For XML mapping, double-click the **CSR_Map** XSLT primitive and enter the name of the map, as shown in Figure 2-56 on page 57, and Click **Next**.

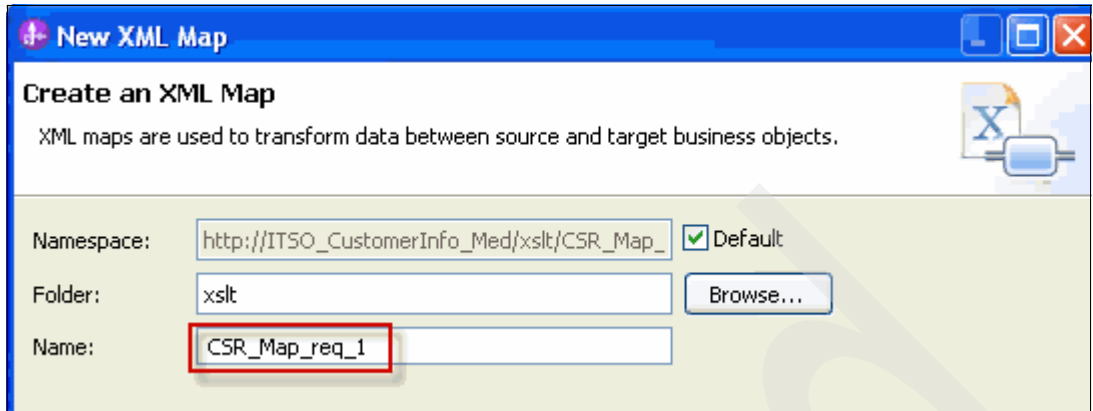


Figure 2-56 Create an XML Map

10. On the Specify Message Types window, in the Message Root field, select **/body**, as shown in Figure 2-57, and click **Finish**.

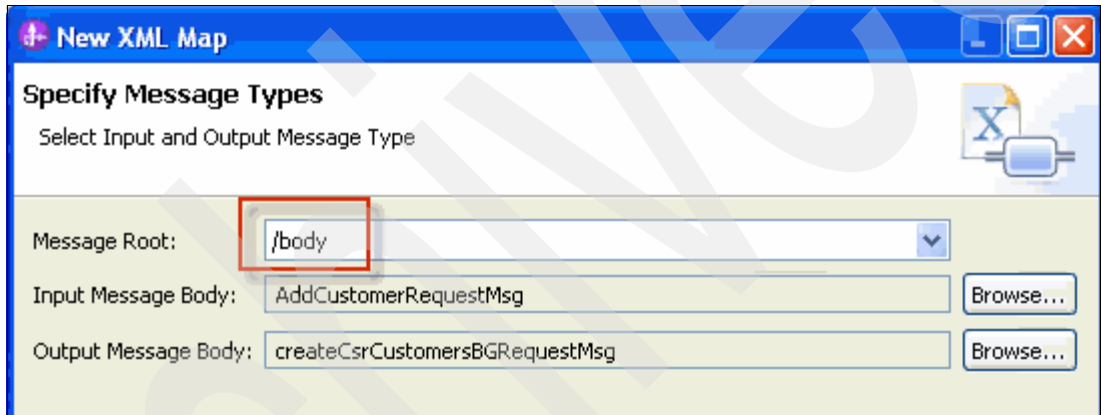


Figure 2-57 Specify Message Types

11. Click the **Auto Map input to output** icon, as shown in Figure 2-58, which automatically maps all similar fields from input to output. Click **Finish**.

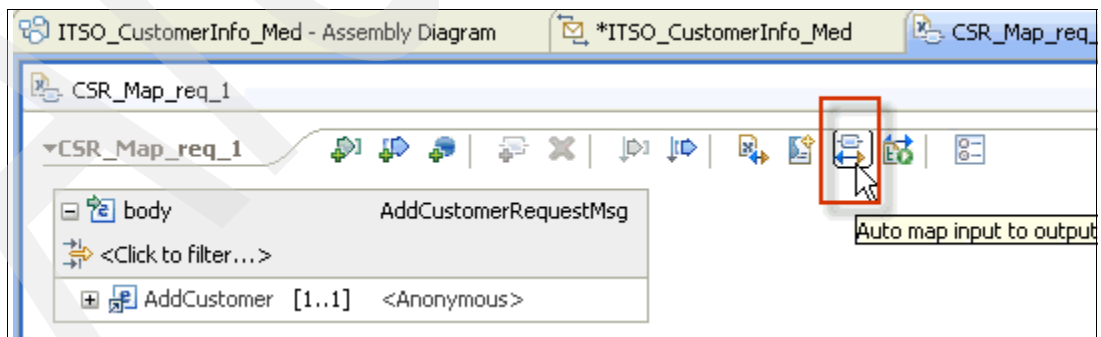


Figure 2-58 XSLT map

12. In the next window, as described in Figure 2-59 on page 58, all similar fields from the source to the target are mapped. If there are any name changes or missed fields for mapping, you can use the **Move** transform to map those fields.



Figure 2-59 XSLT mapping source to target

Next steps: For the XSLT WWW_Map and MST_Map, follow steps 9 through 12 again for the source to target mapping.

13. After finishing the XSLT mapping, we are left with BOMap - **CustID_ResponseMap**, as shown in Figure 2-60 on page 59.

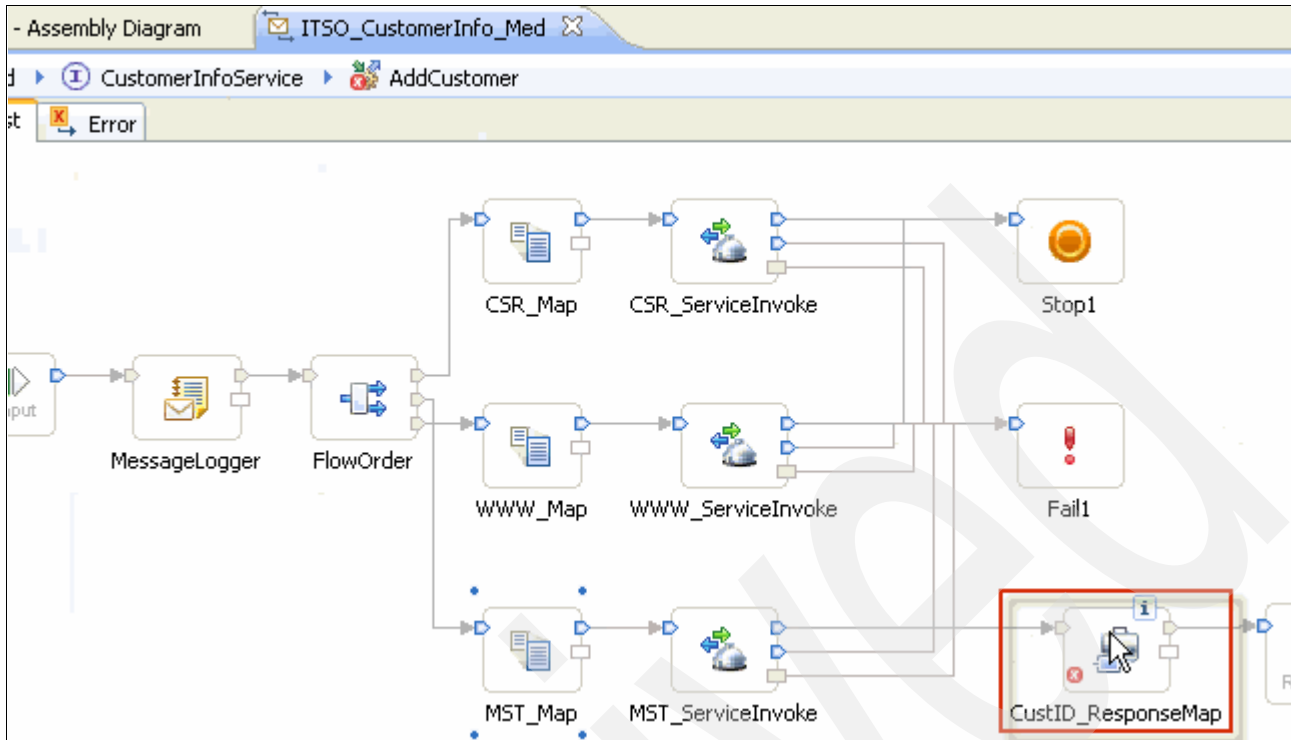


Figure 2-60 Mediation flow

14. Double-click **CustID_ResponseMap** and you see the window, as shown in Figure 2-61. Enter the Name for the business object map, and click **Next**.

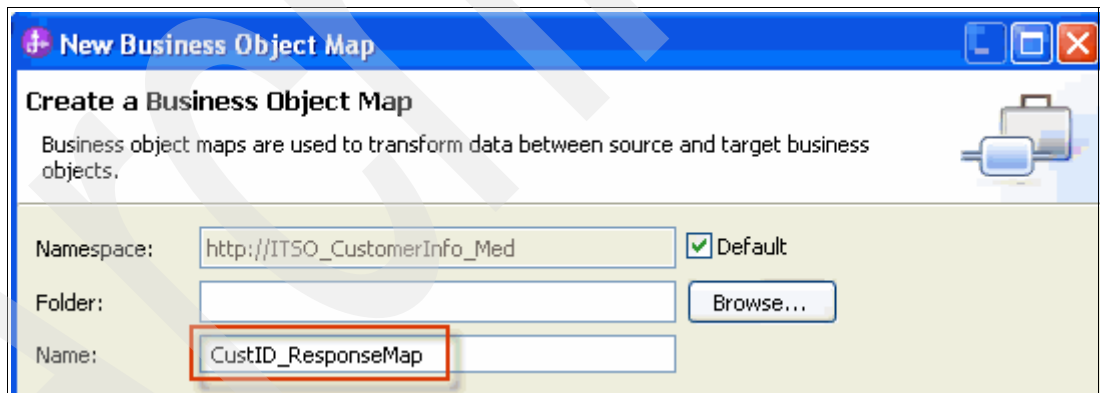


Figure 2-61 New Business Object Map

15. For the Message Root, select **/body**, and click **Finish** (Figure 2-62 on page 60).

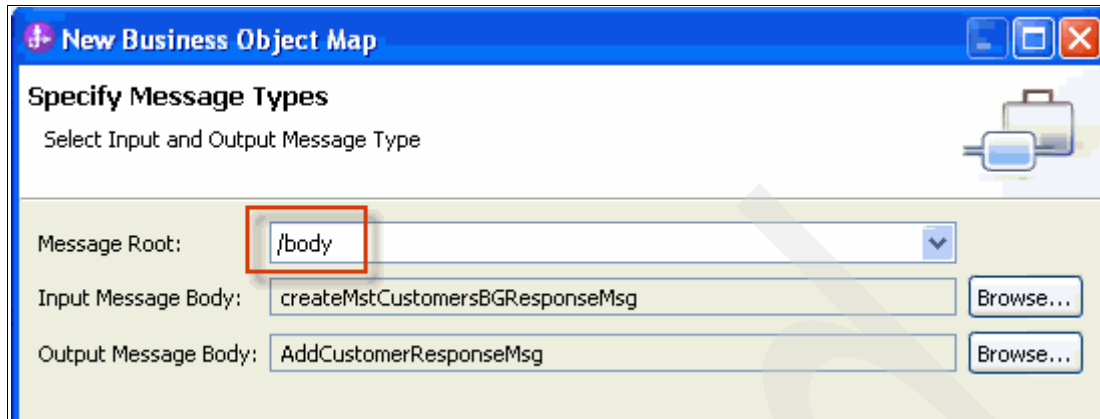


Figure 2-62 Specify Message Types

16. Create a Submap between **createMstCustomersBGOutput** → **AddCustomerResponse** as shown in Figure 2-63.



Figure 2-63 BO map

17. click **Submap** and go to **Properties** → **Details** → **Business object map** → **New**. In the next window, as shown in Figure 2-64, for the Name, type **Response_SubMap**.

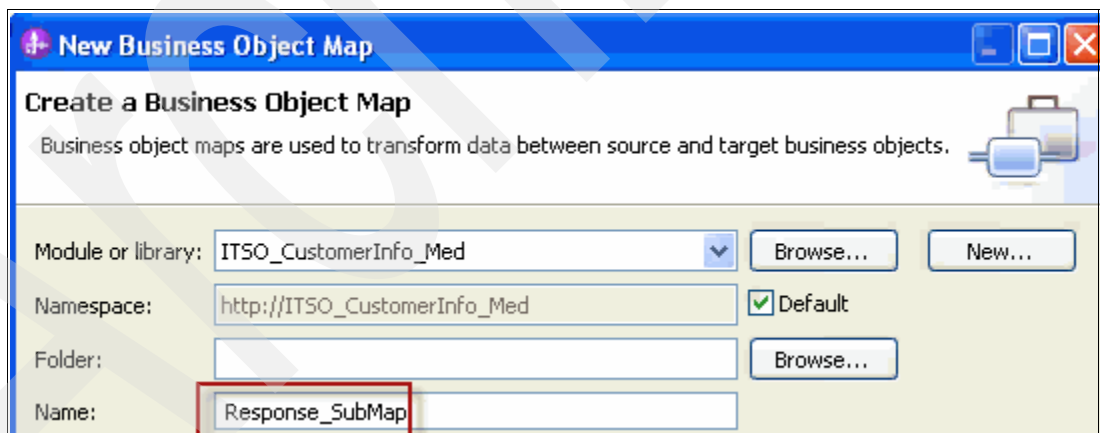


Figure 2-64 Submap in New Business Object Map

18. In the next window, create a **Custom** transform between **id** → **ResponseMessage**, as shown in Figure 2-65 on page 61.

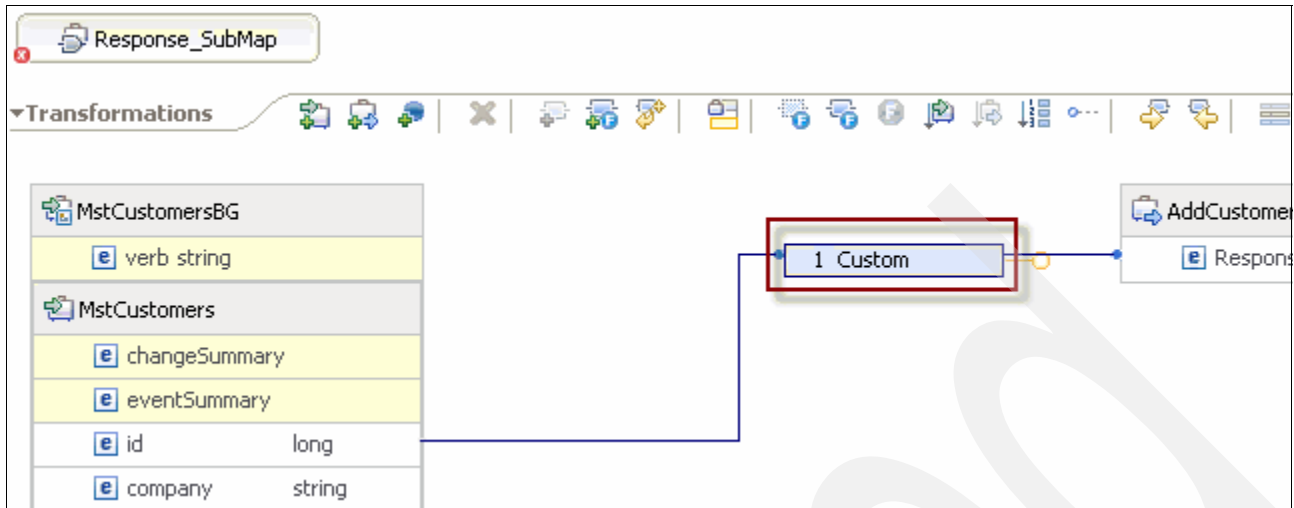


Figure 2-65 BO Map with Custom transform

19. Select the **Custom** transform. Go to **Properties** → **Details** and click the radio button to select the Java perspective. Write the java code, as shown in Figure 2-66, to print the response message after adding the customer details in the DB2 databases.

```
java.lang.Long _customerid = (java.lang.Long)MstCustomersBG_MstCustomers_id;
String temp = Long.toString(_customerid);
AddCustomerResponse_ResponseMessage = "Customer ID is: "+_customerid;
```

Figure 2-66 Java code in mapping to print response message

20. Finally, you see the mediation flow for Add Customer, as shown in Figure 2-67.

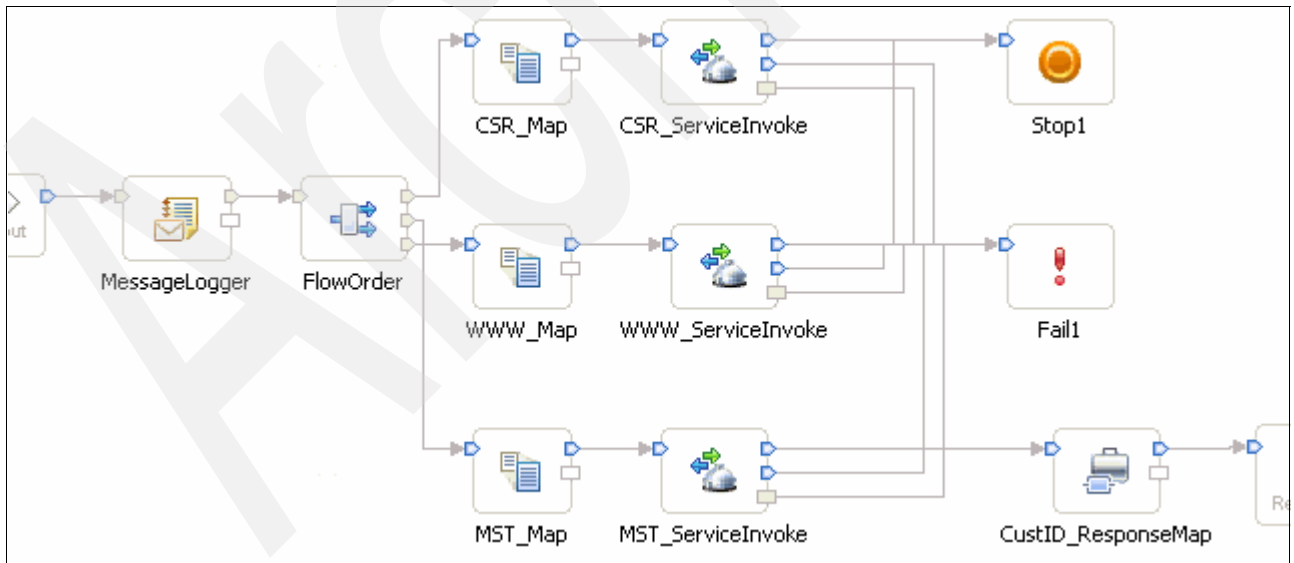


Figure 2-67 Mediation flow for add customers

Updating customer details: Mediation flow development

Update the customer details in the mediation flow development. Perform the following steps:

1. In the mediation flow, go to **Overview**. Right-click **UpdateCustomer** and select **Open Implementation**, as shown in Figure 2-68.

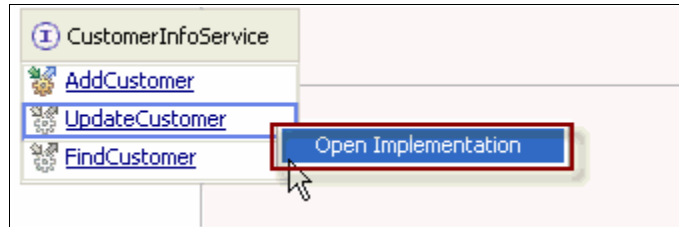


Figure 2-68 Interface representation

2. Select **Service Integration**. In the next window, add **Reference** and **Target Operation** starting with update, as shown in Figure 2-69.

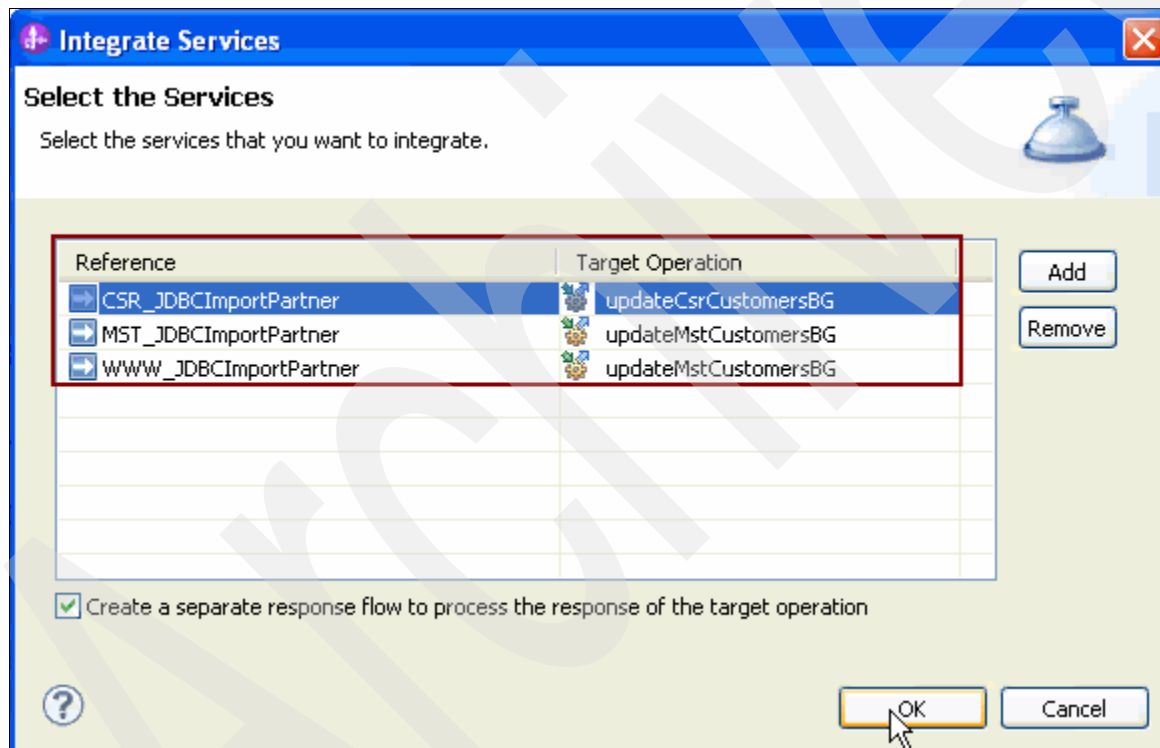


Figure 2-69 Select the services

Steps: For the Update customer details implementation, refer to “Adding customer details: Mediation flow development” on page 53, steps 6 through 17.

For the response message (step 18), as shown in Figure 2-70 on page 63, in the Business Objects map, use **Assign transform** to send the user-defined message, “Customer details are updated successfully in three DB2 databases - CSR, WWW, and MST”.

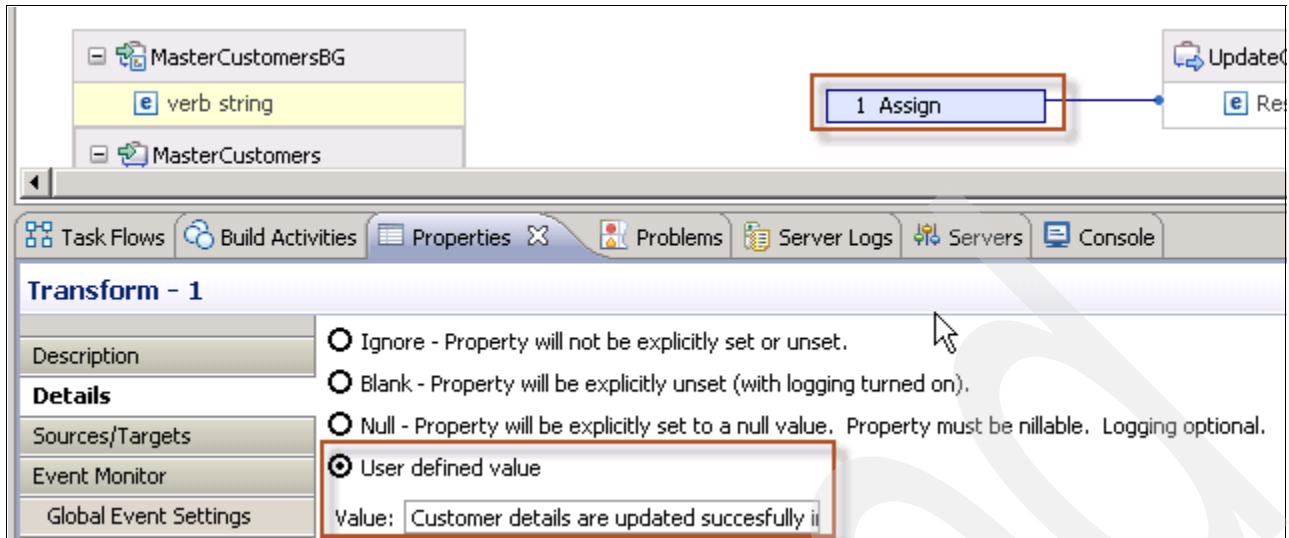


Figure 2-70 Assign transform properties

3. Finally, you see the mediation flow for Update Customer, as shown in Figure 2-71.

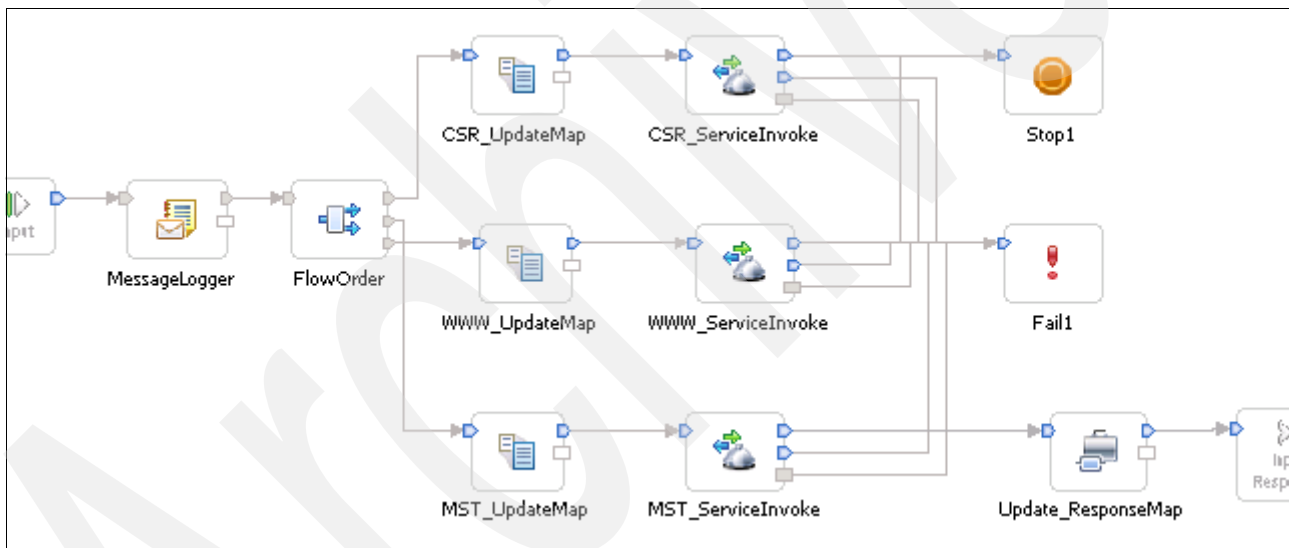


Figure 2-71 Mediation flow for update customer details

Finding customer details: Mediation flow development

We give CustomerID as the input request, and as a response, we get the customer details from the Master DB2 database:

1. In the mediation flow, go to **Overview**, right-click **FindCustomer**, and select **Open Implementation**, as shown in Figure 2-72 on page 64.

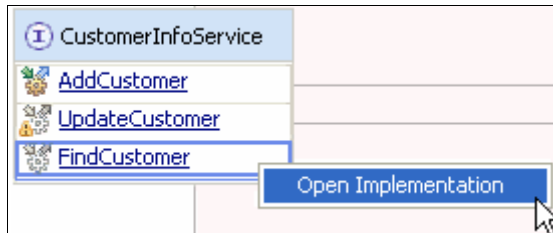


Figure 2-72 Interface representation

2. Click **Blank Mediation Flow**, as shown in Figure 2-73.

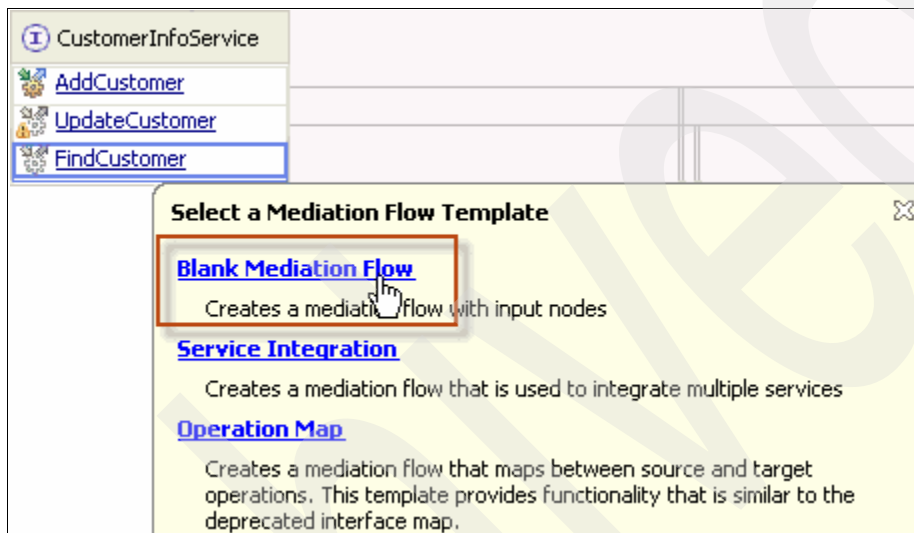


Figure 2-73 Select a Mediation Flow Template

3. In the next window, the mediation flow opens. Click the **CustomerInfoService** interface, as shown in Figure 2-74.

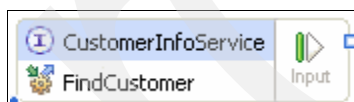


Figure 2-74 CustomerInfoService interface

4. Go to the properties of the **CustomerInfoService** interface, and click **Correlation context** → **Browse** → **CustomerDetails** business object, as described in Figure 2-75 on page 65.

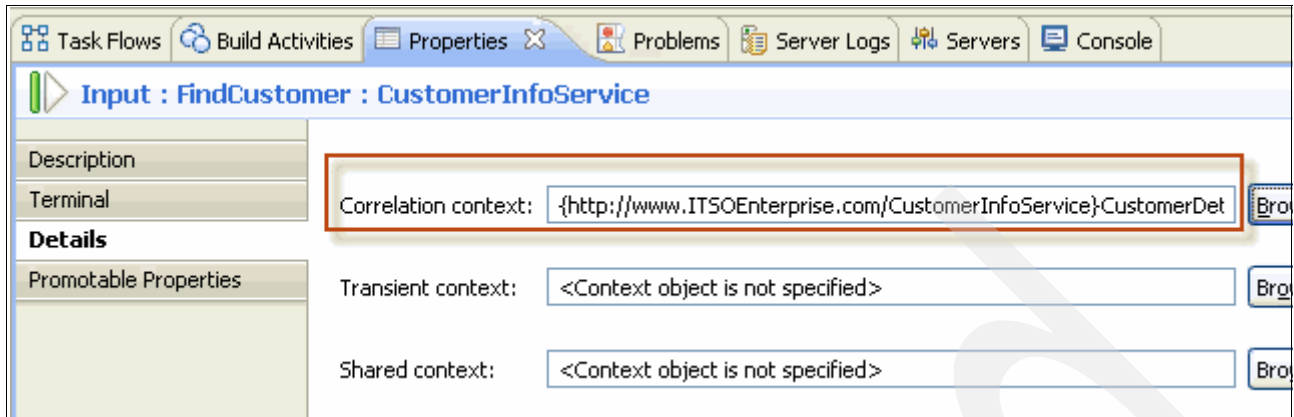


Figure 2-75 CustomerInfoService interface properties

- In the next window, add MessageLogger, Database Lookup, XSLT Map, and Fail primitive in the mediation flow and wire them, as described in Figure 2-76.

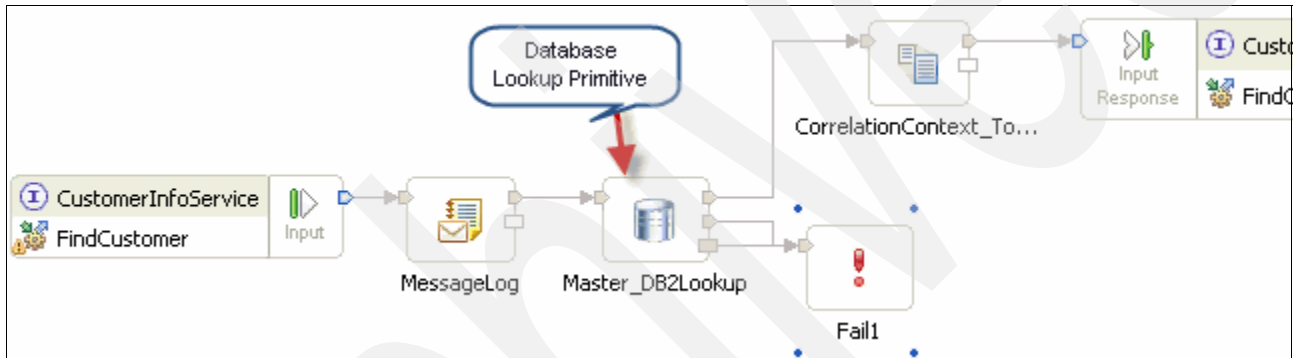


Figure 2-76 Mediation flow: Find customer details

- Click **Master_DB2Lookup** (Database Lookup Primitive), go to **Properties** → **Details**, and enter the details, as described in Figure 2-77 on page 66.

Description	Step 1 - In the message, find the value of the element at the search location. This value																																										
Terminal																																											
Details																																											
Promotable Properties																																											
	Data source:* jdbc/MST																																										
	Table:* MASTER.CUSTOMERS																																										
	Search column:* CUSTOMERID																																										
	Search location:* /body/FindCustomer/CustomerID/CustomerID																																										
	<input type="checkbox"/> Validate input																																										
	Step 2 - From the record retrieved by the search, copy the value of the specified column																																										
	<table border="1"> <thead> <tr> <th>Column</th> <th>Type</th> <th>Target location</th> </tr> </thead> <tbody> <tr> <td>COMPANY</td> <td>xsd:string</td> <td>/context/correlation/company</td> </tr> <tr> <td>CLASS</td> <td>{http://www.ITSOEn...}</td> <td>/context/correlation/class</td> </tr> <tr> <td>C1NAME</td> <td>xsd:string</td> <td>/context/correlation/c1name</td> </tr> <tr> <td>C1EMAIL</td> <td>xsd:string</td> <td>/context/correlation/c1email</td> </tr> <tr> <td>C1PHONE</td> <td>xsd:string</td> <td>/context/correlation/c1phone</td> </tr> <tr> <td>C2NAME</td> <td>xsd:string</td> <td>/context/correlation/c2name</td> </tr> <tr> <td>C2EMAIL</td> <td>xsd:string</td> <td>/context/correlation/c2email</td> </tr> <tr> <td>C2PHONE</td> <td>xsd:string</td> <td>/context/correlation/c2phone</td> </tr> <tr> <td>ADD1</td> <td>xsd:string</td> <td>/context/correlation/add1</td> </tr> <tr> <td>ADD2</td> <td>xsd:string</td> <td>/context/correlation/add2</td> </tr> <tr> <td>ADD3</td> <td>xsd:string</td> <td>/context/correlation/add3</td> </tr> <tr> <td>COUNTRY</td> <td>xsd:string</td> <td>/context/correlation/country</td> </tr> <tr> <td>ZIP</td> <td>xsd:string</td> <td>/context/correlation/zip</td> </tr> </tbody> </table>	Column	Type	Target location	COMPANY	xsd:string	/context/correlation/company	CLASS	{http://www.ITSOEn...}	/context/correlation/class	C1NAME	xsd:string	/context/correlation/c1name	C1EMAIL	xsd:string	/context/correlation/c1email	C1PHONE	xsd:string	/context/correlation/c1phone	C2NAME	xsd:string	/context/correlation/c2name	C2EMAIL	xsd:string	/context/correlation/c2email	C2PHONE	xsd:string	/context/correlation/c2phone	ADD1	xsd:string	/context/correlation/add1	ADD2	xsd:string	/context/correlation/add2	ADD3	xsd:string	/context/correlation/add3	COUNTRY	xsd:string	/context/correlation/country	ZIP	xsd:string	/context/correlation/zip
Column	Type	Target location																																									
COMPANY	xsd:string	/context/correlation/company																																									
CLASS	{http://www.ITSOEn...}	/context/correlation/class																																									
C1NAME	xsd:string	/context/correlation/c1name																																									
C1EMAIL	xsd:string	/context/correlation/c1email																																									
C1PHONE	xsd:string	/context/correlation/c1phone																																									
C2NAME	xsd:string	/context/correlation/c2name																																									
C2EMAIL	xsd:string	/context/correlation/c2email																																									
C2PHONE	xsd:string	/context/correlation/c2phone																																									
ADD1	xsd:string	/context/correlation/add1																																									
ADD2	xsd:string	/context/correlation/add2																																									
ADD3	xsd:string	/context/correlation/add3																																									
COUNTRY	xsd:string	/context/correlation/country																																									
ZIP	xsd:string	/context/correlation/zip																																									

Figure 2-77 Database lookup properties

- In the mediation flow, double-click **CorrelationContext_To_Response_Map** (XSLT Map), click **Next**, as shown in Figure 2-78, and select **/** for the Message Root.

Figure 2-78 Specify Message Types

- In the XSLT Map, map all fields from **context** → **correlation** to **body** → **FindCustomer** → **CustomerDetails**, as described in Figure 2-79 on page 67.

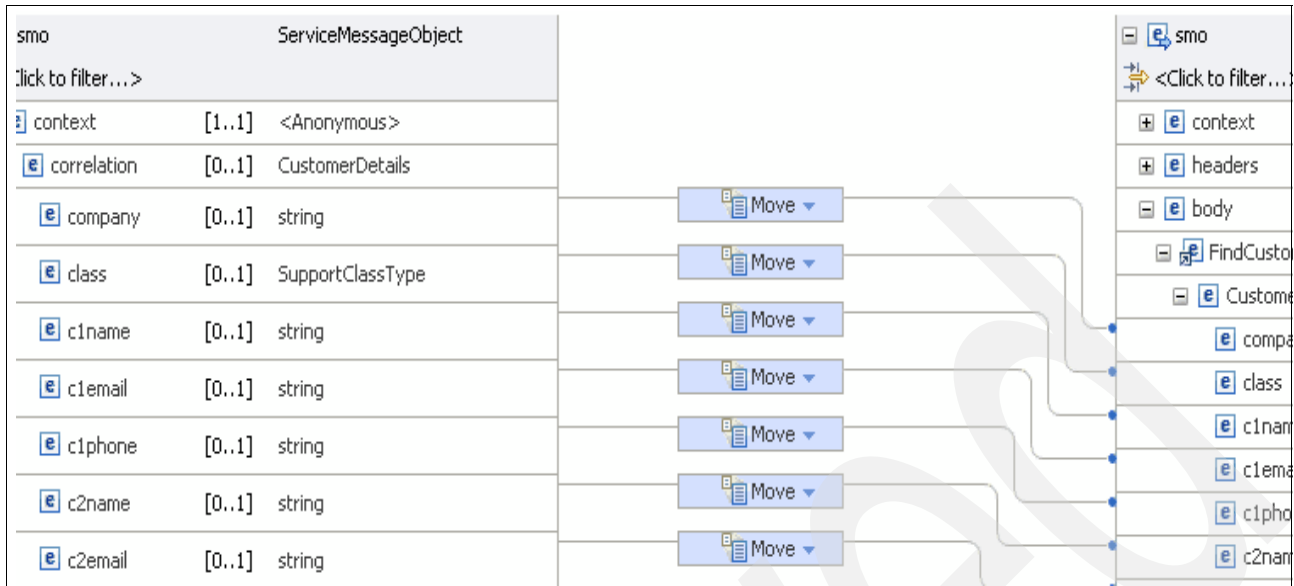


Figure 2-79 XSLT mapping

9. Finally, you see the mediation flow for Find Customer, as shown in Figure 2-80.

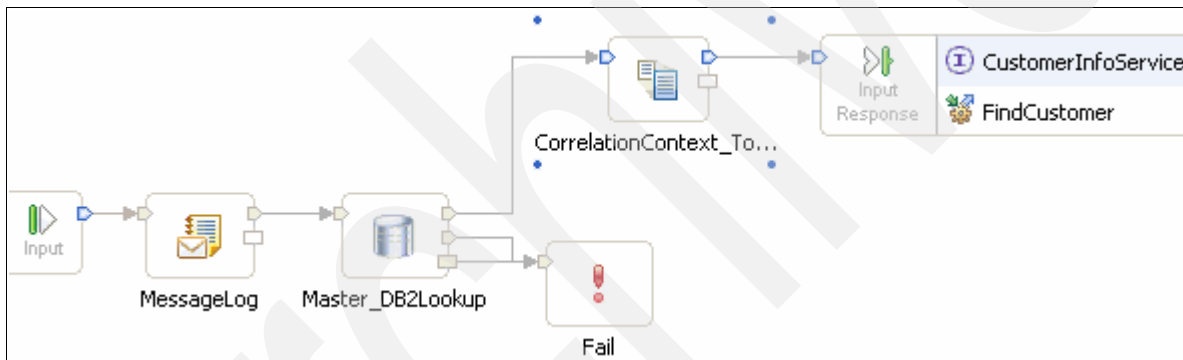


Figure 2-80 Mediation Flow: Find Customer details

2.4.6 Testing the mediation module

This section demonstrates testing the mediation module that we developed for ITSO Enterprise to add, update, and find customer details.

Module deployment

Follow these steps to deploy the mediation module on the ESB Server:

1. Start WebSphere ESB Server v7.5 by clicking **Start**, as shown in Figure 2-81.

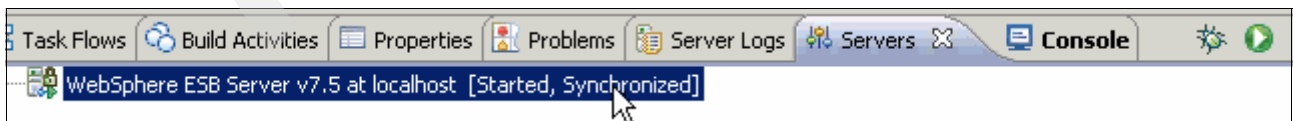


Figure 2-81 Start WebSphere ESB Server v7.5

2. Right-click **WebSphere ESB Server v7.5**, and click **Add and Remove**, as described in Figure 2-82 on page 68.

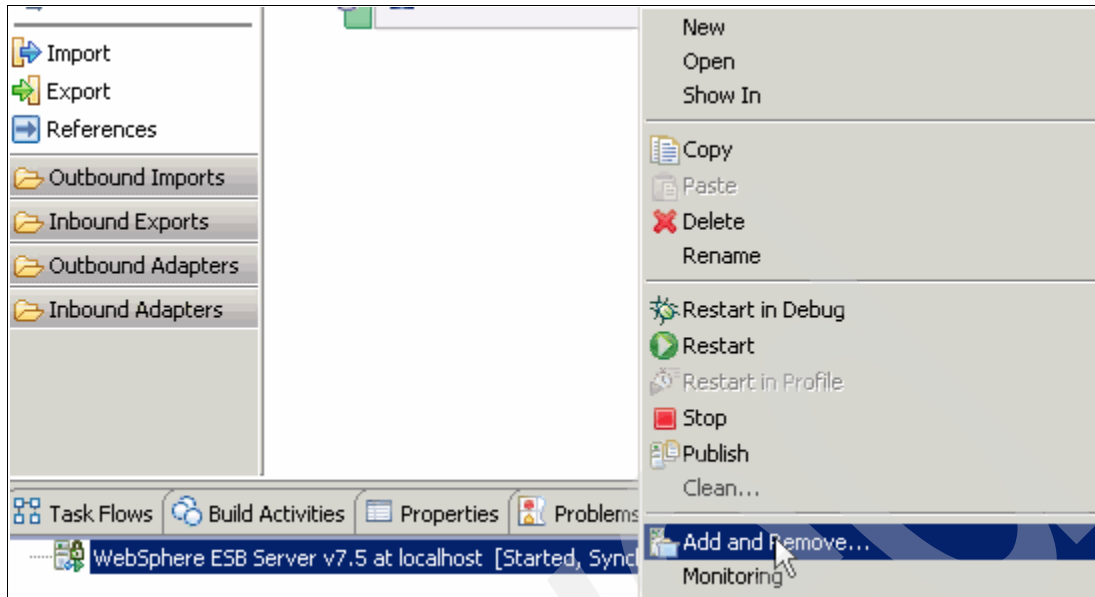


Figure 2-82 Add and Remove mediation module

3. In the next window, select **ITSO_CustomerInfo_MedApp**, click **Add >**, and click **Finish**, as described in Figure 2-83 on page 69.

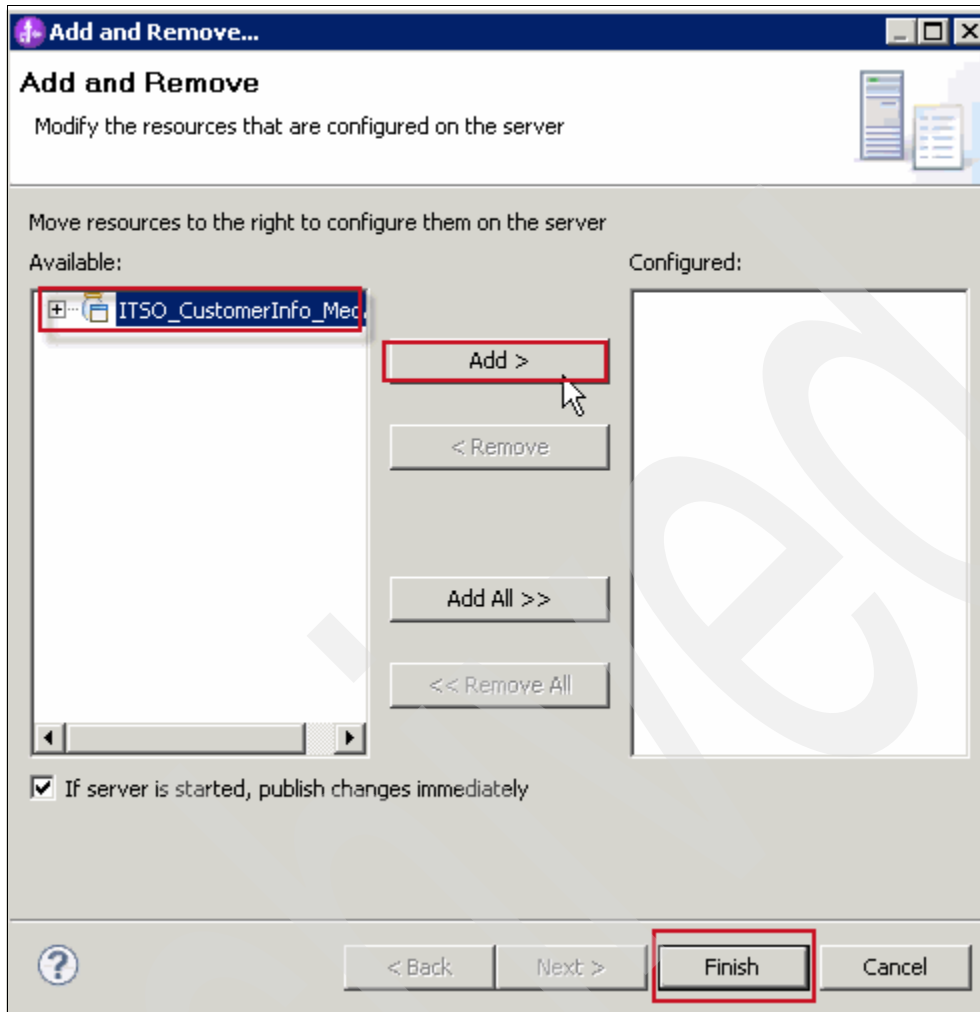


Figure 2-83 Add and Remove

4. After a successful deployment on the WebSphere ESB Server, you see the window that is shown in Figure 2-84.

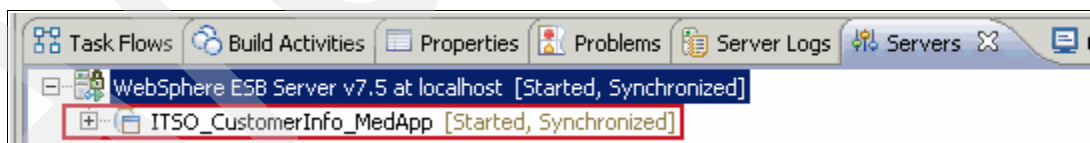


Figure 2-84 After the deployment on the ESB Server

5. Also, as described in Figure 2-85 on page 70, the success message displays in the Console as “Application started: ITSO_CustomerInfoMedApp”.

```

Task Flows Build Activities Properties Problems Server Logs Servers Console
WebSphere ESB Server v7.5 at localhost (WebSphere Application Server v7.0)

I com.ibm.j2ca.dbadapter.core.runtime.DBUtils traceAdapterVersi
I com.ibm.j2ca.dbadapter.core.runtime.DBUtils traceAdapterVersi
I com.ibm.j2ca.dbadapter.core.runtime.DBUtils traceAdapterVersi
I com.ibm.j2ca.dbadapter.core.runtime.DBUtils traceAdapterVersi
grIm I WSVR0049I: Binding javax.resource.cci.ConnectionFactory as
grIm I WSVR0049I: Binding ITSO_CustomerInfo_Med.CSR_JDBCImport_CF
grIm I WSVR0049I: Binding ITSO_CustomerInfo_Med.MST_JDBCImport_CF
grIm I WSVR0049I: Binding ITSO_CustomerInfo_Med.WWW_JDBCImport_CF
I com.ibm.ws.webcontainer.webapp.WebGroupImpl WebGroup SRVE0169
nCor I SessionContextRegistry getSessionContext SESNO176I: Will crea
I com.ibm.ws.webcontainer.servlet.ServletWrapper init SRVE0242I
xten I WWS7037I: The /sca/CustomerInfoServiceExport URL pattern
ner I com.ibm.ws.wsewebcontainer.VirtualHost addWebApplication SRVE0
nSpe I J2CA0291I: Application ITSO_CustomerInfo_MedApp#ITSO_Custome
e I [:] CWSIVO777I: A connection to messaging engine storeNodeC
nSpe I J2CA0523I: The Message Endpoint for ActivationSpec sca/ITSO
onMg A WSVR0221I: Application started: ITSO_CustomerInfo_MedApp
onUn A WSVR0191I: Composition unit WebSphere:cuname=ITSO_CustomerI

```

Figure 2-85 Console

Testing: Add customer details

Use Web service testing utilities to test the endpoint to the ITSO_CustomerInfo web service. Example 2-1 shows the AddCustomer Request that we POST through this test utility.

Example 2-1 SOAP request message: Add customer details

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:cus="http://www.ITSOEnterprise.com/CustomerInfoService">
  <soapenv:Header/>
  <soapenv:Body>
    <cus:AddCustomer>
      <CustomerDetails>
        <company>IBM</company>
        <class>gold</class>
        <c1name>CustomerCare</c1name>
        <c1email>CustomerCare@ibm.com</c1email>
        <c1phone>1234567890</c1phone>
        <c2name>?</c2name>
        <c2email>?</c2email>
        <c2phone>?</c2phone>
        <add1>IBM Headquarters</add1>
        <add2>?</add2>
        <add3>?</add3>
        <country>USA</country>
        <zip>?</zip>
      </CustomerDetails>
    </cus:AddCustomer>
  </soapenv:Body>
</soapenv:Envelope>

```

Response message

As a result, we see the response message as the Customer ID number, which is generated from the CSR, WWW, and MST DB2 databases, as described in Example 2-2.

Example 2-2 SOAP Response message: Add customer details

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <p:AddCustomerResponse
xmlns:p="http://www.ITSOEnterprise.com/CustomerInfoService">
      <ResponseMessage>Customer ID is: 10107</ResponseMessage>
    </p:AddCustomerResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Response message: We send the AddCustomer SOAP Request by calling the ITSO_CustomerInfoService web service, which inserts the customer details into three DB2 databases: CSR, WWW, and MST. The CustomerID is generated automatically from the DB2 database. As a result, we get the CustomerID as a response message.

Updating the customer details

As we discussed earlier in the ITSO Enterprise scenario problem, after registering, the customer decides to update its address.

Example 2-3 shows the UpdateRequest that we send through the test utility.

Example 2-3 SOAP Request Message: Update Customer details

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:cus="http://www.ITSOEnterprise.com/CustomerInfoService">
  <soapenv:Header/>
  <soapenv:Body>
    <cus:UpdateCustomer>
      <UpdateCustomerDetails>
        <CustomerID>10107</CustomerID>
        <company>IBM</company>
        <class>gold</class>
        <c1name>CustomerCare</c1name>
        <c1email>CustomerCare@ibm.com</c1email>
        <c1phone>1234567890</c1phone>
        <c2name>?</c2name>
        <c2email>?</c2email>
        <c2phone>?</c2phone>
        <add1>IBM-ITSO</add1>
        <add2>?</add2>;
        <add3>?</add3>
        <country>USA</country>
        <zip>?</zip>
      </UpdateCustomerDetails>
    </cus:UpdateCustomer>
  </soapenv:Body>
</soapenv:Envelope>
```

Response Message

As a result, we see the response message that is shown in Example 2-4 on page 72.

Example 2-4 SOAP Response Message: Update Customer details

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <p:UpdateCustomerResponse xmlns:p="http://www.ITS0Enterprise.com/CustomerInfoService">
      <ResponseMessage>Customer details are updated succesfully in 3 Databases
-CSR,WWW,MST</ResponseMessage>
    </p:UpdateCustomerResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Response message: We send an UpdateCustomer SOAP Request by calling the ITS0_CustomerInfoService web service, which updates the customer address in three DB2 databases: CSR, WWW, and MST. As a result, we get the success response message.

Finding the customer details

If the customers on the web portal or the Customer Sales Representatives on the CSR system want to find a customer's details, they can send a SOAP request with the CustomerID, as described in Example 2-5.

Example 2-5 SOAP Request message: Find Customer details

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:cus="http://www.ITS0Enterprise.com/CustomerInfoService">
  <soapenv:Header/>
  <soapenv:Body>
    <cus:FindCustomer>
      <CustomerID>
        <CustomerID>10107</CustomerID>
      </CustomerID>
    </cus:FindCustomer>
  </soapenv:Body>
</soapenv:Envelope>
```

Response message

As a result, we get a response with all the customer details in SOAP response message, as shown in Example 2-6.

Example 2-6 SOAP Response Message: Find Customer details

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <io8:FindCustomerResponse
xmlns:io4="wsdl.http://www.ITS0Enterprise.com/CustomerInfoService"
xmlns:io5="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:io3="http://www.ibm.com/xmlns/prod/websphere/mq/sca/6.0.0"
xmlns:io2="http://www.ibm.com/websphere/sibx/smo/v6.0.1"
xmlns:io="http://www.w3.org/2003/05/soap-envelope"
xmlns:xs4xs="http://www.w3.org/2001/XMLSchema" xmlns:io9="http://ITS0_Lib"
xmlns:io8="http://www.ITS0Enterprise.com/CustomerInfoService"
xmlns:io7="http://www.w3.org/2005/08/addressing"
xmlns:io6="http://www.ibm.com/xmlns/prod/websphere/http/sca/6.1.0">
      <CustomerDetails>
```

```
<company>IBM</company>
<class>gold</class>
<cname>CustomerCare</cname>
<c1email>CustomerCare@ibm.com</c1email>
<c1phone>1234567890</c1phone>
<c2name>?</c2name>
<c2email>?</c2email>
<c2phone>?</c2phone>
<add1>IBM-ITSO</add1>
<add2>?</add2>
<add3>?</add3>
<country>USA</country>
<zip>?</zip>
</CustomerDetails>
</io8:FindCustomerResponse>
</soapenv:Body>
</soapenv:Envelope>
```

Response message: We send the FindCustomer SOAP Request by calling the ITSO_CustomerInfoService web service, which finds the customer details from the MASTER DB2 database. As a result, we get all the customer details for the given CustomerID with the SOAP Response message.

2.5 Summary

In this chapter, we described how ITSO Enterprise adopted a Smart SOA approach using IBM WebSphere ESB to integrate all three of the systems that hold the data: Master (DB2), WWW (DB2), and CSR (DB2). Also, we discussed how WebSphere ESB provides flexible service mediation and hosting for integrating applications and data as services to enable the development of an SOA.

WebSphere ESB can connect virtually any business application and supports common connectivity patterns, such as simple proxy, service gateway, service translation, and service selection, as well as mediation policies for dynamic SOA solutions. Also, WebSphere ESB brings consistency to point-to-point connectivity diversity, reducing IT complexity while improving business agility.

WebSphere ESB integrates well with WebSphere Service Registry and Repository, supporting a common business space environment for web-based widgets to SOA users. In next chapter, we discuss how we can integrate WebSphere ESB with WebSphere Service Registry and Repository and demonstrate how we can make use of available services within ITSO Enterprise and the governance of services.

Archived

Service Enablement pattern: Enterprise Content Management System

This chapter describes the Service Enablement pattern aspect of WebSphere Adapter for Enterprise Content Management and IBM FileNet. As explained in Chapter 1, “Introduction” on page 1, ITSO Enterprise wants to build applications to store and view its suppliers’ legal contract documents on an on-demand basis. ITSO Enterprise is keen on building this solution on WebSphere Enterprise Service Bus (WebSphere ESB) using IBM WebSphere Adapter for Enterprise Content Management.

This chapter covers the following sections and gives you a complete story of the end-to-end implementation process:

- ▶ 3.1, “Introduction to the business scenario” on page 76
- ▶ 3.2, “Assembly Diagram” on page 76
- ▶ 3.3, “Benefits” on page 77
- ▶ 3.4, “Technical implementation” on page 77

3.1 Introduction to the business scenario

After spending many years in the local market, ITSO Enterprise thinks it needs to globalize its business by entering other countries. However, ITSO Enterprise wants to establish a standard mechanism within the organization to manage its suppliers' legal contract documents. The process involves storing scanned physical contract documents that have been signed between ITSO Enterprise and its global suppliers on the FileNet system and reviewing the documents on a periodic basis for the renewal, rejection, or approval of these documents.

3.1.1 Scenario problem

ITSO Enterprise has these challenges while implementing this business scenario:

- ▶ Because ITSO Enterprise is an environmentally friendly company, it needs to be compliant with the *Paperwork Reduction Act*, thus reducing the paperwork by storing documents electronically on FileNet.
- ▶ ITSO Enterprise needs to make documents available on demand to selected teams and the legal department only, because the documents are highly confidential and contain crucial information, such as agreed-to price, contract terms, tenure, and so on.

3.1.2 Scenario solution

ITSO Enterprise has created a two-step process, which uses WebSphere Adapter for Enterprise Content Management and exposes FileNet content as an appropriate business service per the service-oriented architecture (SOA) model:

- ▶ Because most of the ITSO Enterprise supplier data is stored in its central supplier management system, ITSO Enterprise uses the supplier data with FTP connectivity to upload documents to FileNet when the legal office signs a new contract with a supplier.
- ▶ ITSO Enterprise has built an internal web application for querying the contract documents based on various input criteria. This web application uses the queryDocuments service, as explained in 3.4.4, "Building a web application on the retrieveAllDocuments service to query FileNet documents" on page 135. In this fictitious scenario, the ITSO Enterprise central supplier management system at one end generates the required supplier information. The legal team retrieves that information and copies it over to an FTP location along with a scanned version of the legal contract documents.

3.2 Assembly Diagram

We implemented this scenario using the following IBM products:

- ▶ IBM Integration Designer v7.5
<http://www-01.ibm.com/software/integration/integration-designer>
- ▶ IBM WebSphere ESB Server v7.5
<http://www-01.ibm.com/software/integration/wsesb>
- ▶ IBM WebSphere Adapter for Enterprise Content Management v7.5
http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r5mx/topic/com.ibm.wsadapters.jca.ecm.doc/doc/stbp_ecm_welcome.html

- ▶ IBM FileNet Content Engine v4.5.1 or v4.6
<http://publib.boulder.ibm.com/infocenter/p8docs/v4r5m1/index.jsp>

Figure 3-1 shows a higher-level block diagram for the scenario implementation.

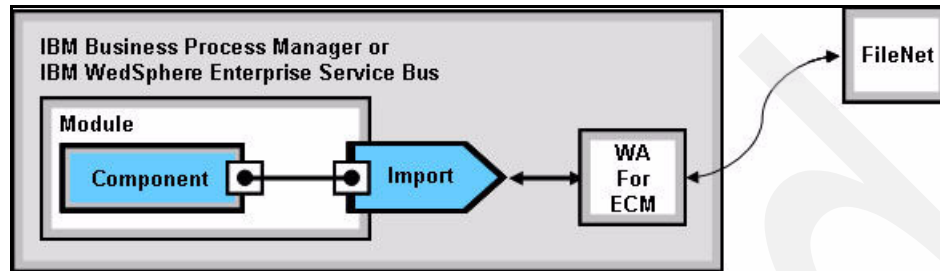


Figure 3-1 Higher-level block diagram of the solution involving WebSphere Adapter for Enterprise Content Management

3.3 Benefits

In general, *service integration* refers to integrating various applications that run on separate platforms using separate protocols.

In our scenario, ITSO Enterprise has a central supplier management system, which contains its core supplier information. ITSO Enterprise wants to use FileNet to store the scanned version of the corresponding legal supplier contract information.

To integrate these two systems, ITSO Enterprise needs to build an application around them to get real-time data from both systems, for example, to query documents based on specific criteria.

IBM offers a few products that support ESB functionalities. Based upon clients' business needs, clients can choose any of the ESB products to implement service integration in their IT environments.

In this scenario, we use IBM WebSphere ESB as the ESB product to integrate the central supplier management system and the FileNet content management system.

3.4 Technical implementation

We describe the steps that are required for the actual implementation of the proposed business solutions in four sections:

- ▶ Create a custom document class to manage legal contract documents electronically on FileNet.
- ▶ Create business services using WebSphere Adapter for Enterprise Content Management to service-enable FileNet's content.
- ▶ Build an FTP interface for one of the business services that is exposed via WebSphere Adapter for Enterprise Content Management to store documents on the FileNet system.
- ▶ Build a web application on top of another business service to query documents from FileNet based on specific criteria.

Figure 3-2 explains all possible business services of FileNet exposed via WebSphere Adapter for Enterprise Content Management. Two of them, *createDocument* and *queryDocuments*, are consumed via separate protocol technologies: FTP and Web (HTTP). You can perform the entire implementation for this business solution in a day or two because of the highly flexible and extendable architecture of WebSphere ESB technology.

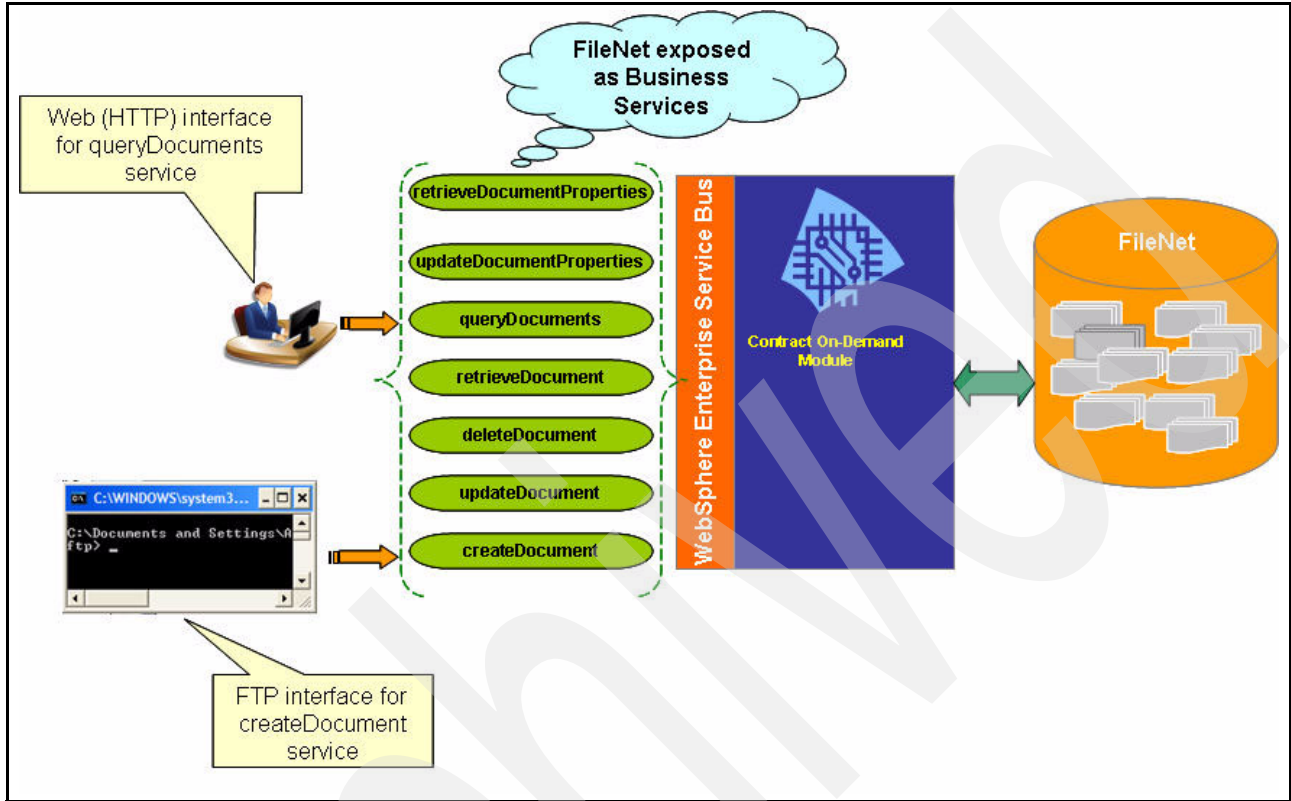


Figure 3-2 FileNet business services exposed via WebSphere Adapter for Enterprise Content Management

3.4.1 Configuring FileNet with custom Document Class type to manage legal contract documents electronically

This section takes you through a step-by-step process to create a new custom Document Class type called *Supplier Contract* on FileNet and to add the appropriate business properties to it. Follow these steps:

1. To start the implementation, log in to FileNet using the FileNet Enterprise Manager Administration Tool with the appropriate login credentials.

Log in: Check with your FileNet administrator or the FileNet product documentation if you have trouble logging in to the FileNet content engine using the FileNet Enterprise Manager Administration tool.

2. After you have successfully logged in to FileNet's content engine, you see a list of all the repositories in the content engine. Right-click the target repository where you want to create a new document class type and choose **New** → **Class**, as shown in Figure 3-3 on page 79.

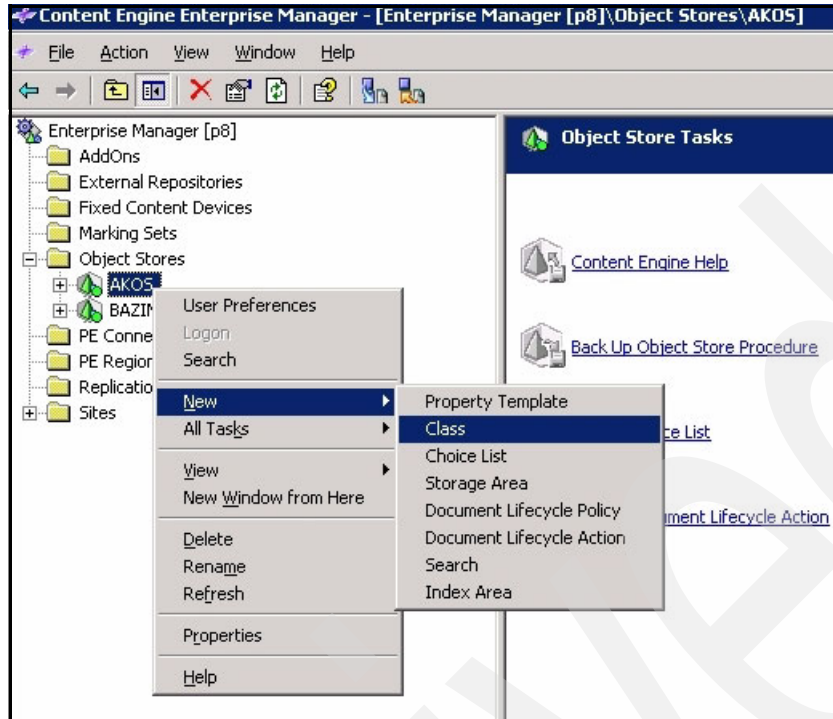


Figure 3-3 Create new document class

3. The previous step opens a window where you enter the document class name. Enter Supplier Contract. See Figure 3-4.

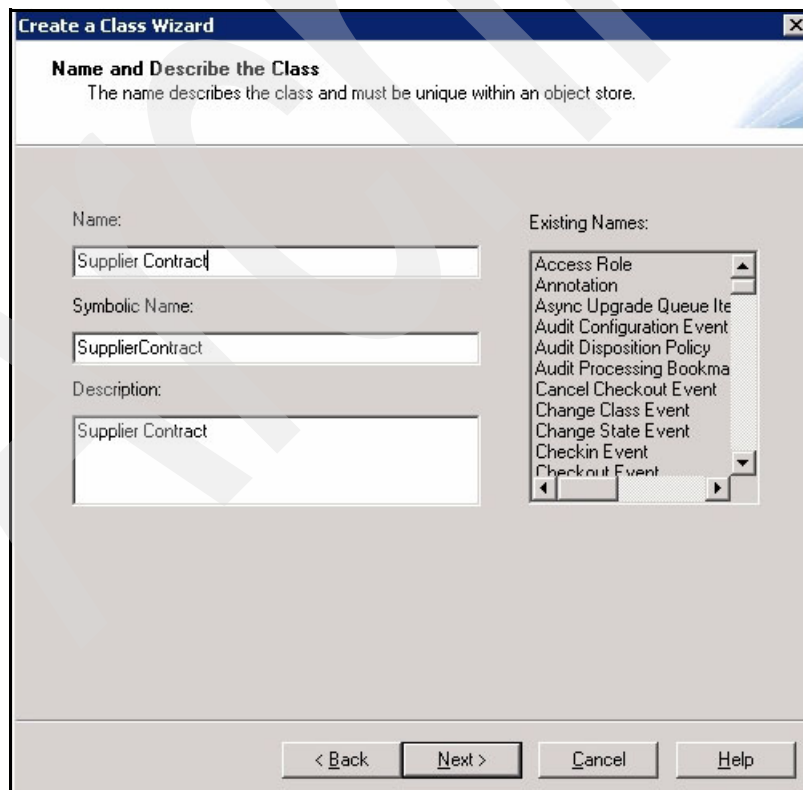


Figure 3-4 Enter a valid name for the Document Class: Supplier Contract

- Next, select the appropriate superclass type for the document class that we are creating. Extend it from the Document superclass, because the nature of the entity for which we will use it is supplier contract documents (Figure 3-5).

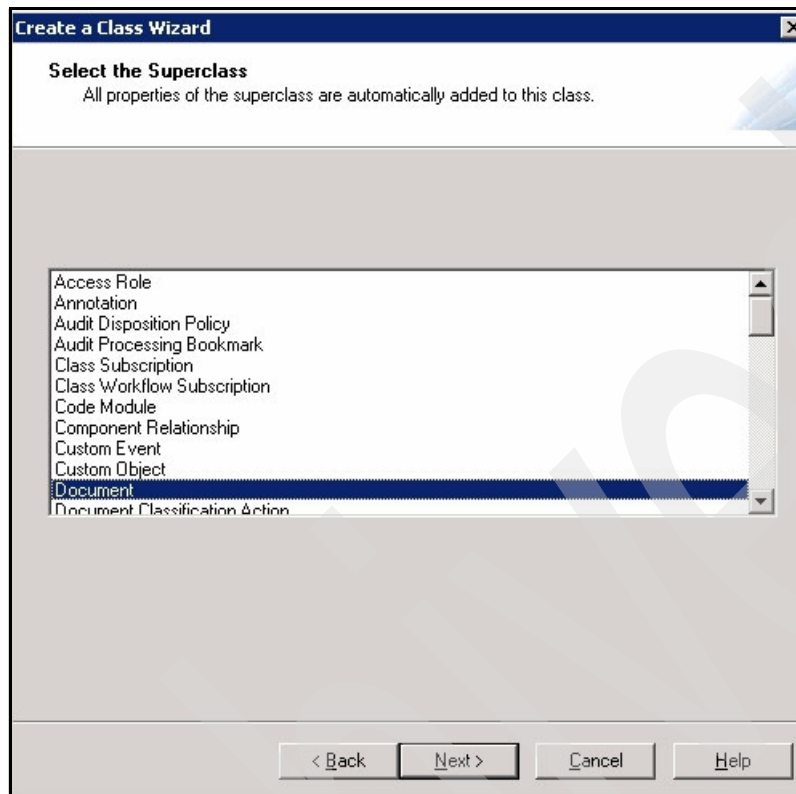


Figure 3-5 Select Document as the superclass for the document class type that we will create

Note: For more information about various types of classes that are supported by the FileNet system, refer to the IBM FileNet product documentation:

<http://publib.boulder.ibm.com/infocenter/p8docs/v4r5m1/index.jsp>

- Click **Next**.
- It is the time to add custom properties to the document class. *Properties* are the individual values that describe an object. You can add as many properties as your business needs. You can also define properties with various characteristics, such as required, multi-cardinality, and so on. For our business case, we require only a few properties. After you select the superclass for the document class that we are creating, you see the next page of the wizard where you can add the properties. Figure 3-6 on page 81 is the first step of the process.

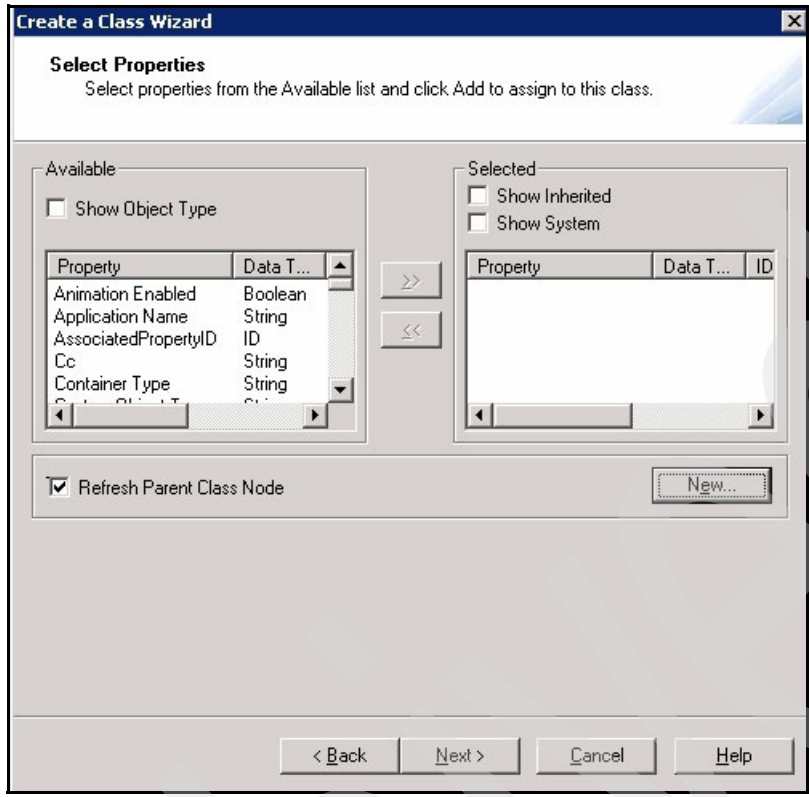


Figure 3-6 Page to configure document properties

7. Click **New** to start creating properties. The window that is shown in Figure 3-7 opens, where for the Name of the property, you enter Primary Legal.

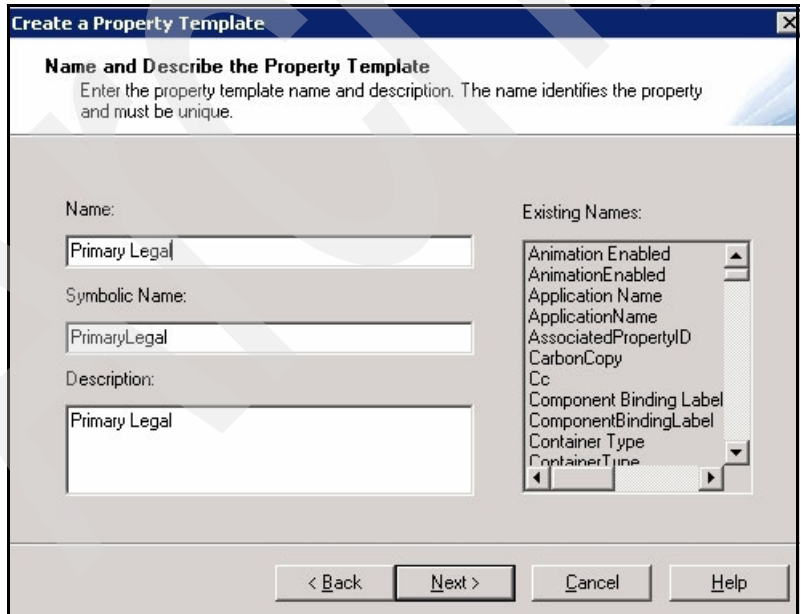


Figure 3-7 Enter the name for the property

8. On the next page, select **String** for the data type for the property, as shown in Figure 3-8 on page 82, and click **Next**.

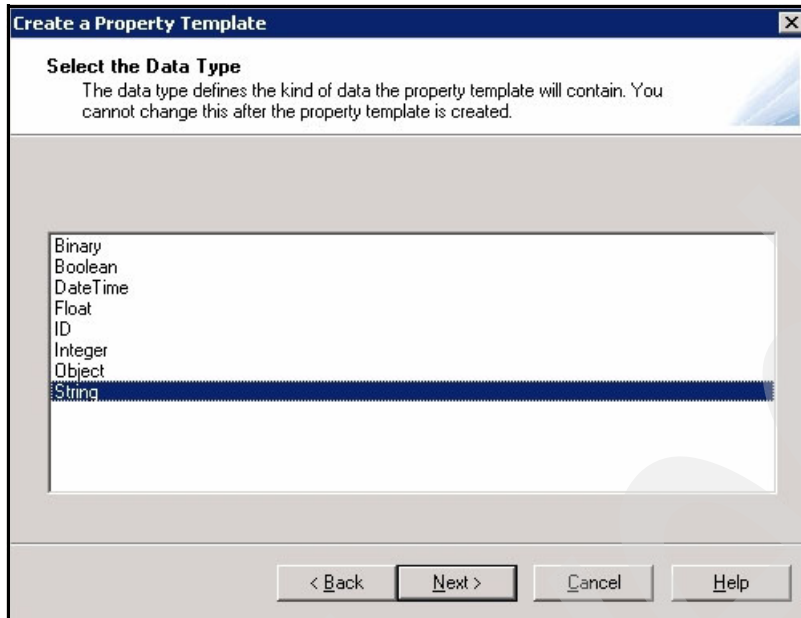


Figure 3-8 Select the data type for the property

9. On the next window, assign a choice list of the property that is being created. Because this property is a simple type, we do not need to select this option. See Figure 3-9.

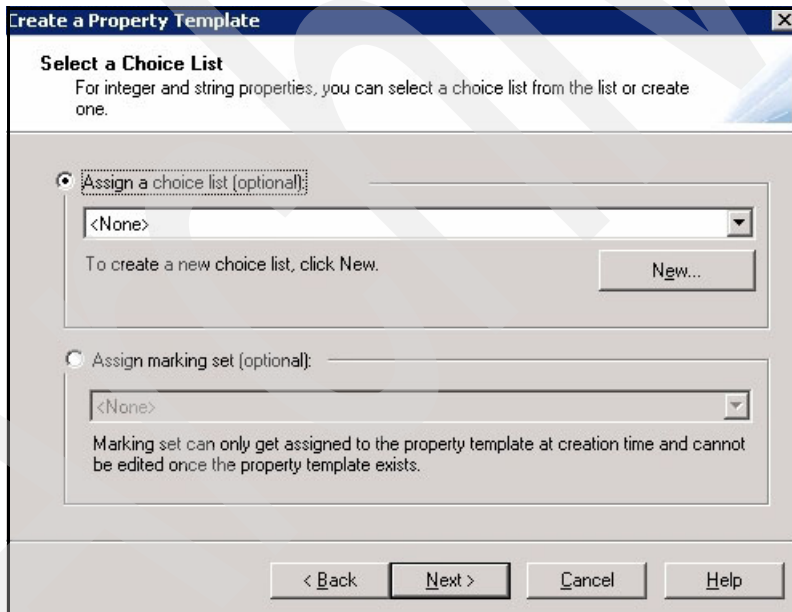


Figure 3-9 Optionally assign a choice list for the property

10. Click **Next**.

11. The page that is shown in Figure 3-10 on page 83 helps you set the type of cardinality for the property. It can be either Single or Multi. Choose **Single**. The Primary Legal property denotes the name of the primary legal officer. This field can only have one value at a time.

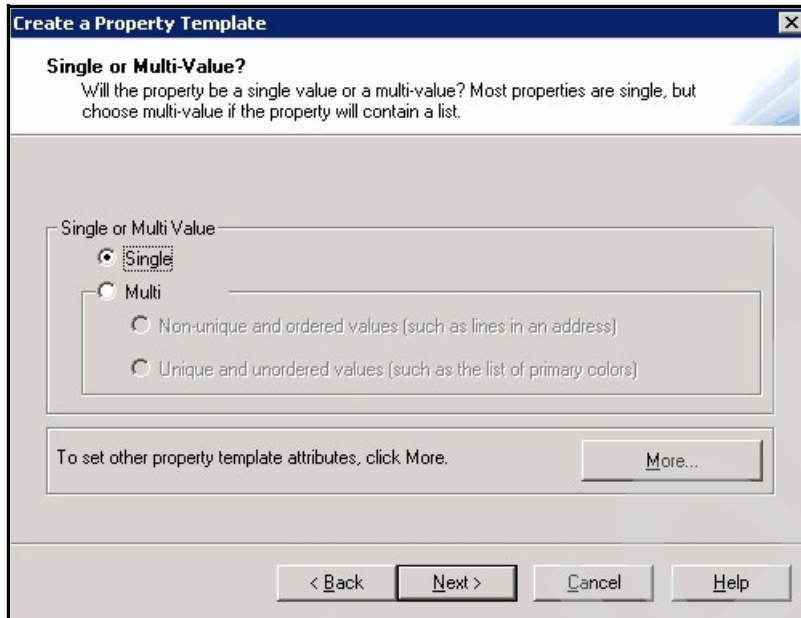


Figure 3-10 Choose the type of property as either single or multi

12. Click **Next** to complete the property configuration.

13. You have completed the process of creating a property. A summary page, as shown in Figure 3-11, appears as a result.

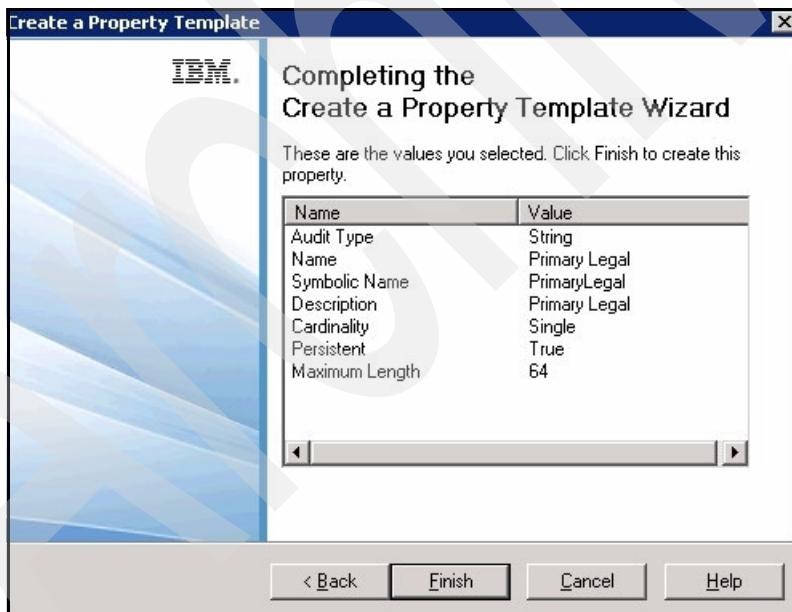


Figure 3-11 Summary page for property creation

14. Repeat the property creation steps to add a few more properties, which are listed in Table 3-1 on page 84, to the document class.

Table 3-1 Required properties for the document class

Property name	Data type	Cardinality
Secondary Legal	String	Single
Supplier ID	String	Single
Expiry Date	DateTime	Single

15. The wizard page that is shown in Figure 3-12 appears after you successfully add the required properties to the document class. Click **Next**.

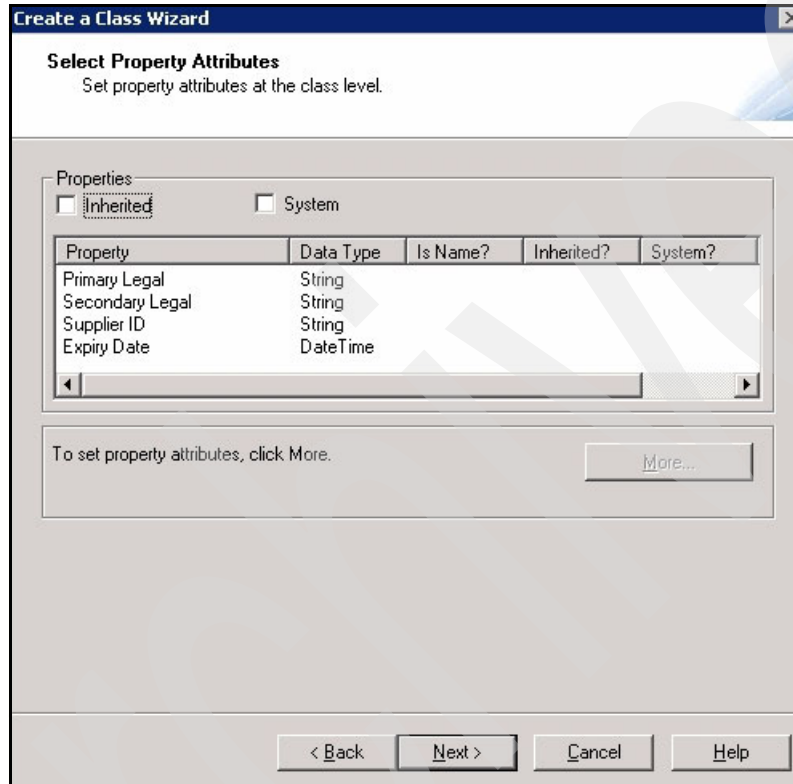


Figure 3-12 Final list of properties added to the document class

16. After you have added all the required properties to the document class, you must configure the settings for inheritance and auditing. Because the document class to be created does not require these settings, proceed with the default values for them. Figure 3-13 on page 85 shows the wizard page to configure auditing for the document class.

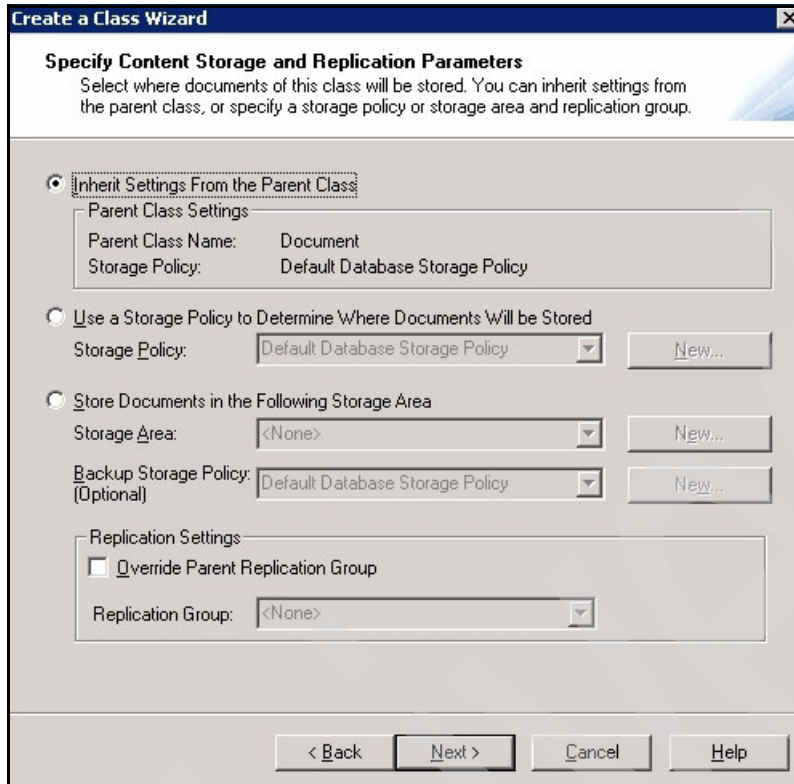


Figure 3-13 Inheritance settings to be configured from the chosen parent class

17. Keep the default values and click **Next**. A wizard page, as shown in Figure 3-14 on page 86, appears to configure auditing.

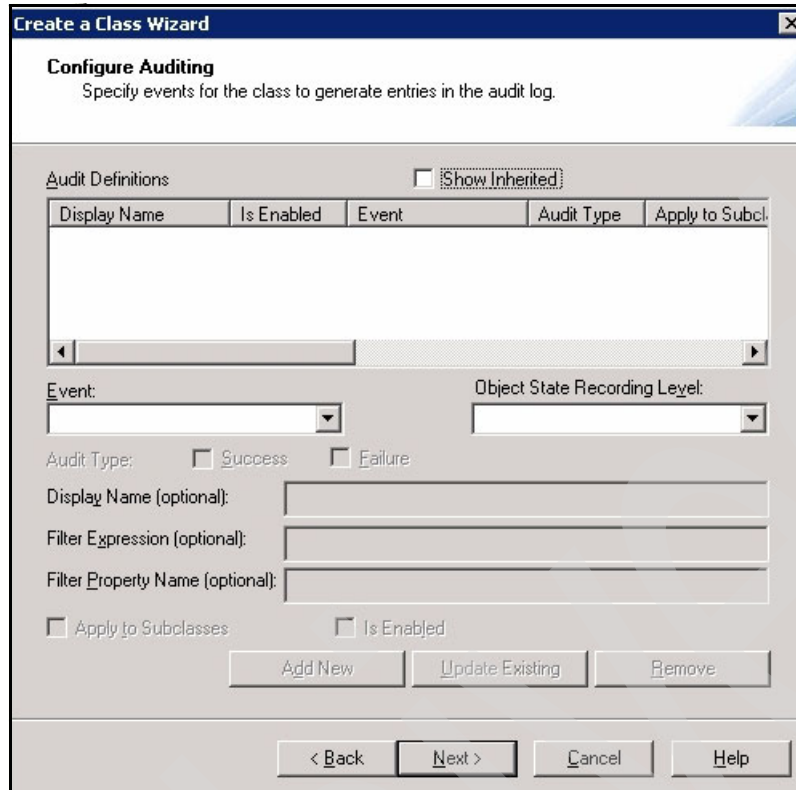


Figure 3-14 Auditing definitions for the document class to be created

18. Keep the default values and click **Next**.

19. A summary page, as shown in Figure 3-15 on page 87, is displayed as confirmation for the successful creation of the document class.

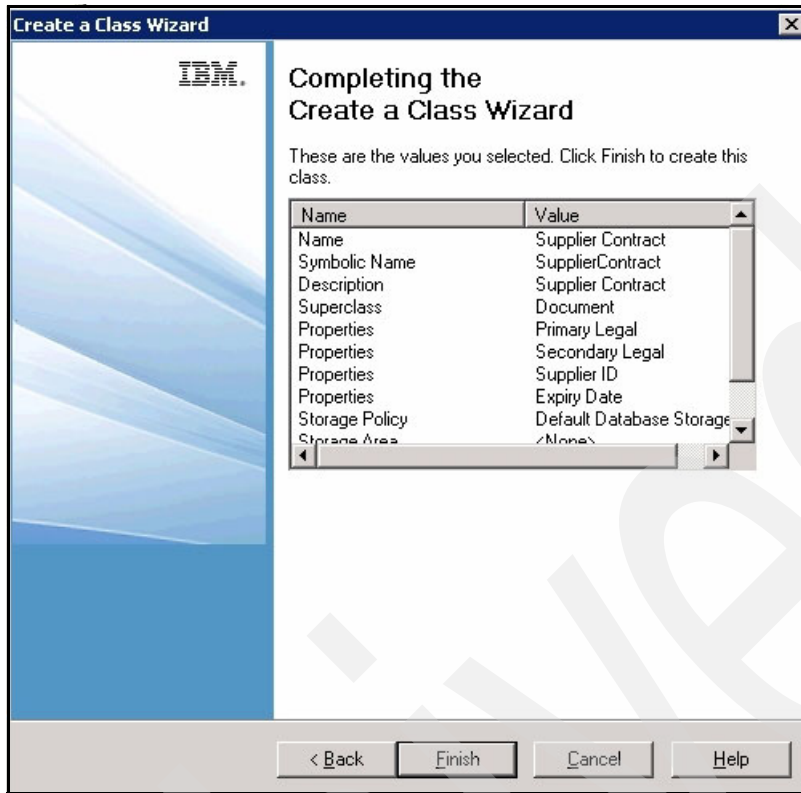


Figure 3-15 Summary page for class creation wizard

20. Click **Finish**.

21. Right-click the object store, **AKOS**, and refresh the repository. Expand the Document Class node to see the newly created document class named Supplier Contract (Figure 3-16).

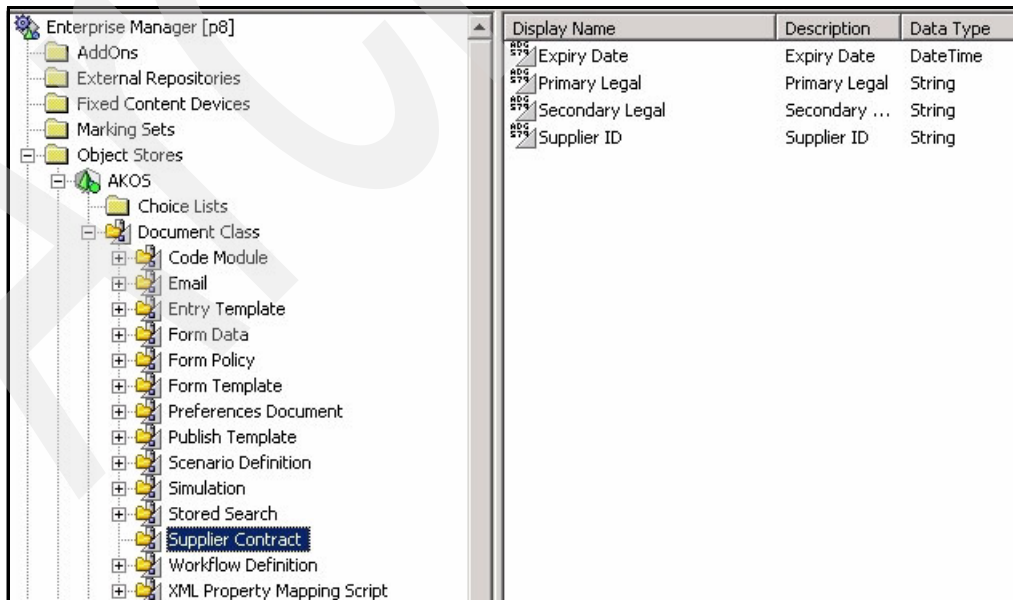


Figure 3-16 Supplier Contract document class successfully created on the repository AKOS

3.4.2 Using WebSphere Adapter for Enterprise Content Management to service-enable FileNet content

This section takes you through a step-by-step process to service-enable the FileNet content, which is the new document class named Supplier Contract that you have created. This process requires IBM Integration Designer v7.5 and IBM WebSphere Enterprise Service Bus v7.5. Follow these steps:

1. Launch IBM Integration Designer and create an workspace to store your project artifacts.
- 2.

For more information: Refer to the IBM Integration Designer product documentation for more details about how to install, set up, and use the product. You can access the product documentation at this website:

<http://publib.boulder.ibm.com/infocenter/esbsoa/wesbv7r5/index.jsp?topic=/com.ibm.wbpm.wid.main.doc/welcome.html>

3. After you successfully create the workspace, the Business Integration perspective launches automatically. Right-click the **Business Integration** project explorer view and create a new mediation module, as shown in Figure 3-17.

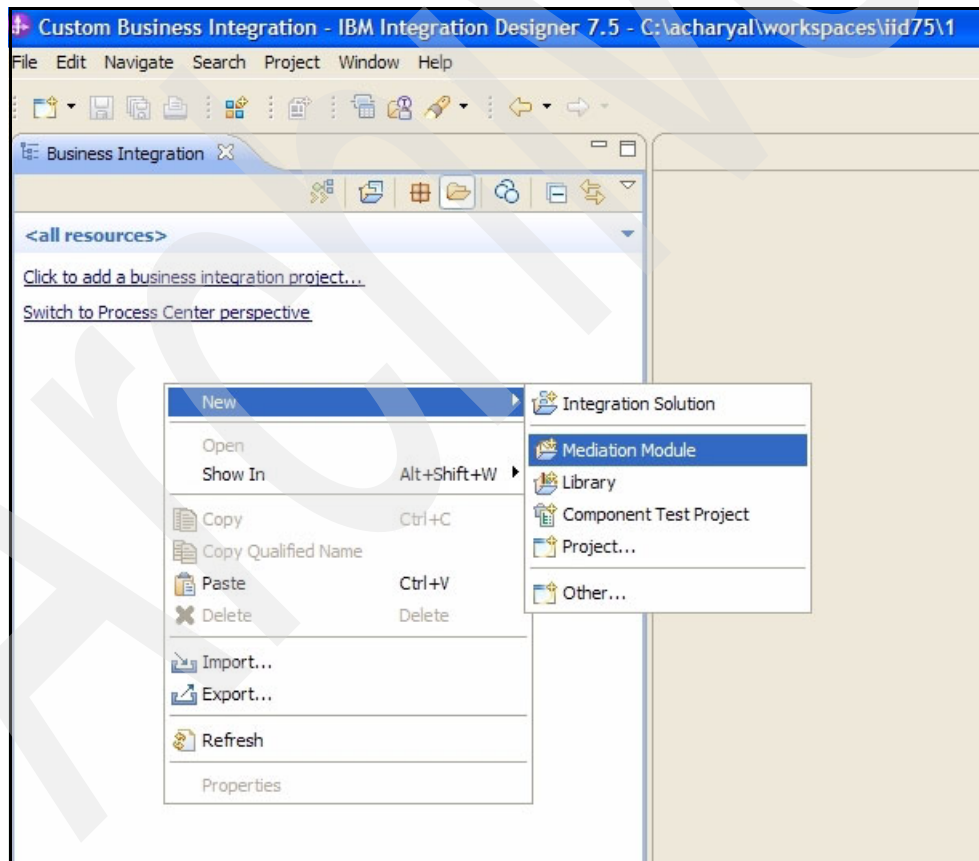


Figure 3-17 Create a new mediation module to store project artifacts

4. On the next page, enter a name for the mediation module, as shown in Figure 3-18 on page 89. In our case, the mediation module name is `itso.legal.contract`.

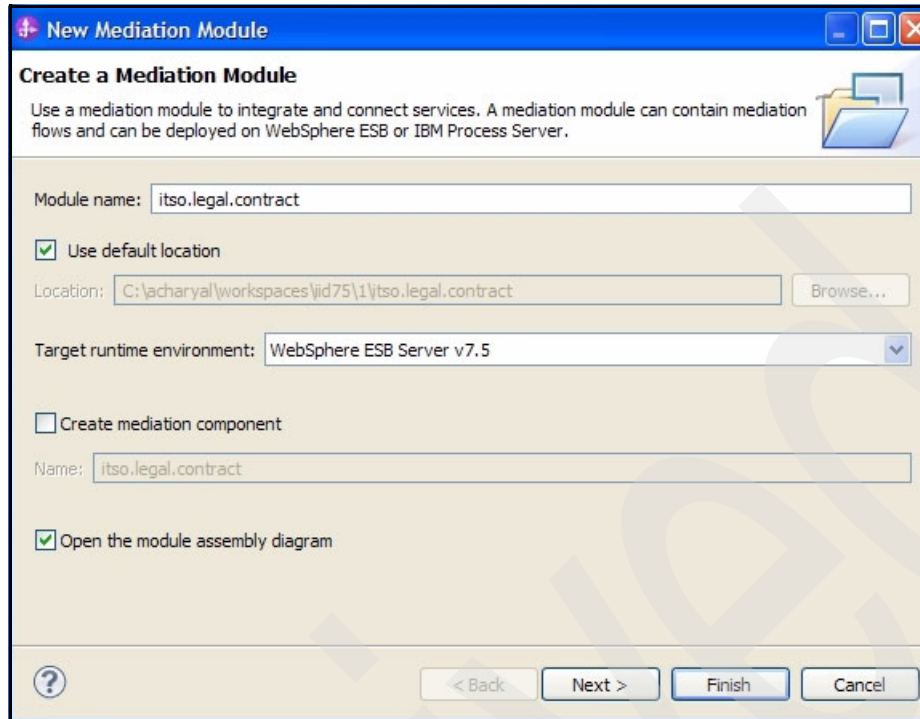


Figure 3-18 Enter a name for the mediation module

- The next window allows you to choose the type of business object parsing mode for this mediation module. Choose **Lazy parsing**, because it provides better performance compared to the Eager parsing mode. See Figure 3-19.

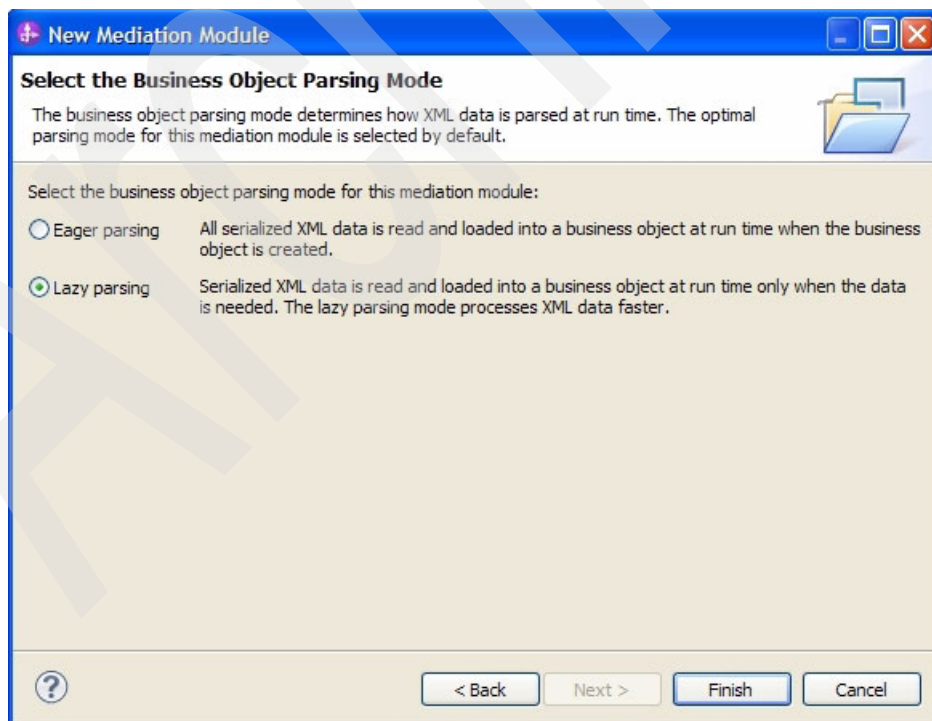


Figure 3-19 Choose Lazy parsing for the parsing mode for the mediation module

You have completed the creation of the mediation module. The next step is to use the IBM WebSphere Adapter to service-enable the FileNet content. This interesting exercise involves connecting to the target ECM, FileNet, and graphically selecting the resource, Supplier Contract document class, to be service-enabled for the business application to call.

6. For the first step toward Service Enablement, launch the External Service wizard of the IBM WebSphere Adapter for Enterprise Content Management by dragging and dropping the **Enterprise Content Management** control from the **Outbound Adapters** palette to the Assembly Diagram, as shown in Figure 3-20.

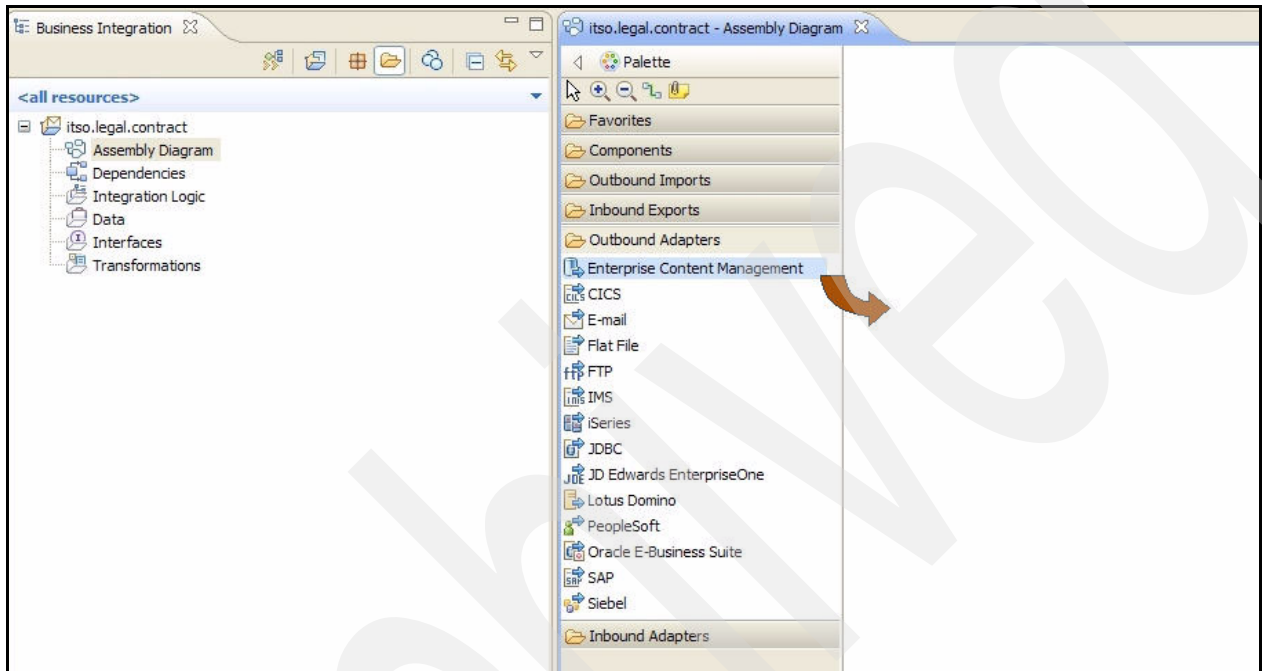


Figure 3-20 Drag and drop the ECM control onto the Assembly Diagram

7. This action triggers the discovery process of the adapter. On the next window, select the version of the adapter that you want to use for the service enablement of the FileNet application. Because there is only one version of adapter available now, a window similar to Figure 3-21 on page 91 opens.

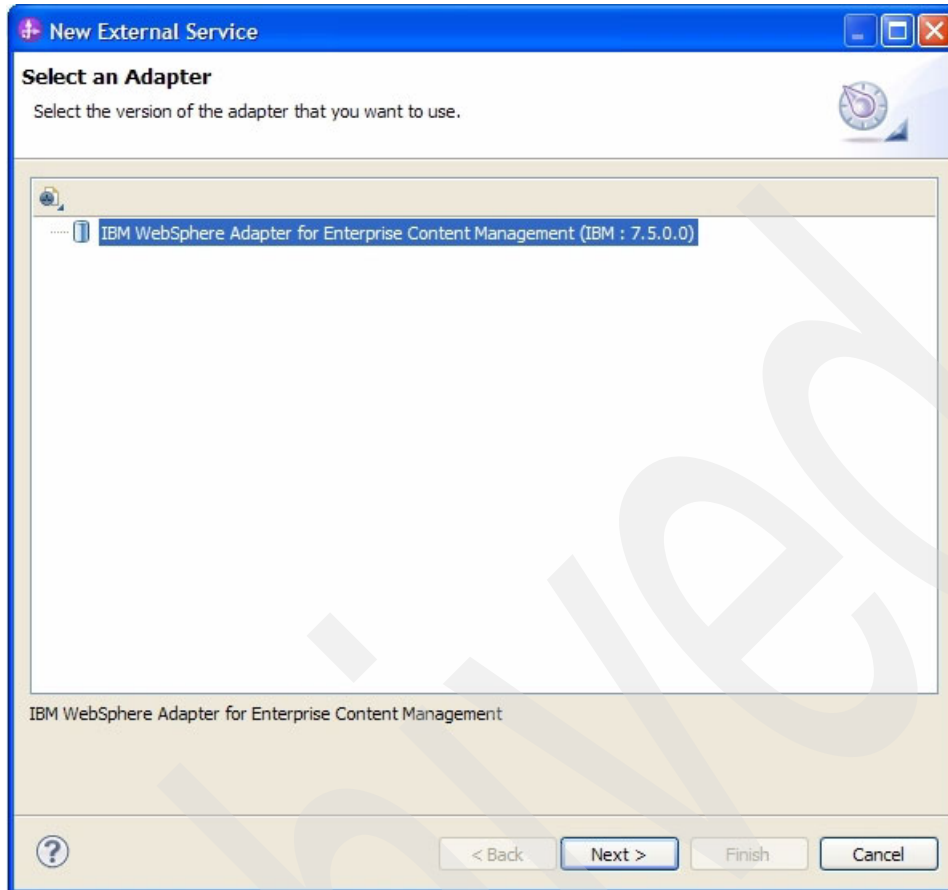


Figure 3-21 Select the adapter for the FileNet discovery process

8. Choose the listed adapter and click **Next**. The next window allows you to import the adapter's binary, a rar file, into the workspace. Keep the default values, unless you want to import the file with another name, and click **Next**. A window similar to Figure 3-22 on page 92 opens.

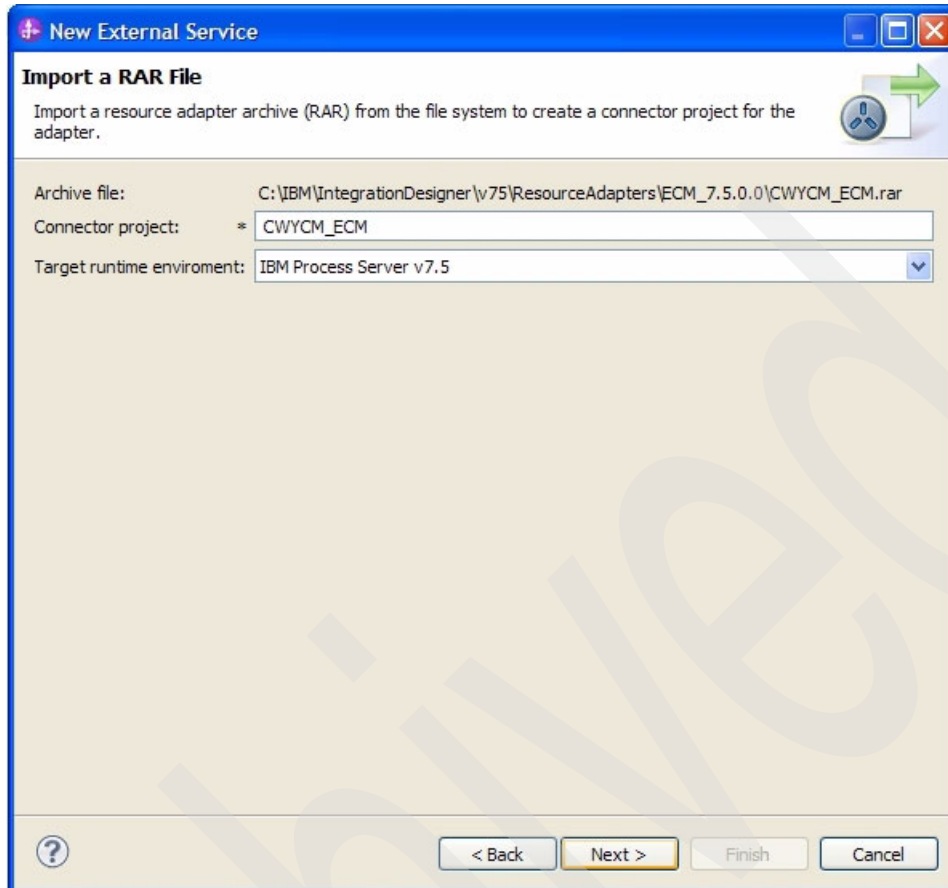


Figure 3-22 Importing the adapter's binary RAR file into the workspace

9. Configure the dependency files that are required for the adapter to work.

Important: If you have installed IBM Integration Designer with the test environment enabled, step 9 does not apply to you.

10. Select the **com.ibm.ws.runtime.jar** file from the WebSphere Enterprise Service Bus installation directory, and click **Next**. See Figure 3-23 on page 93.

Dependency file location: The location of the dependency file varies for each operation's system environment.

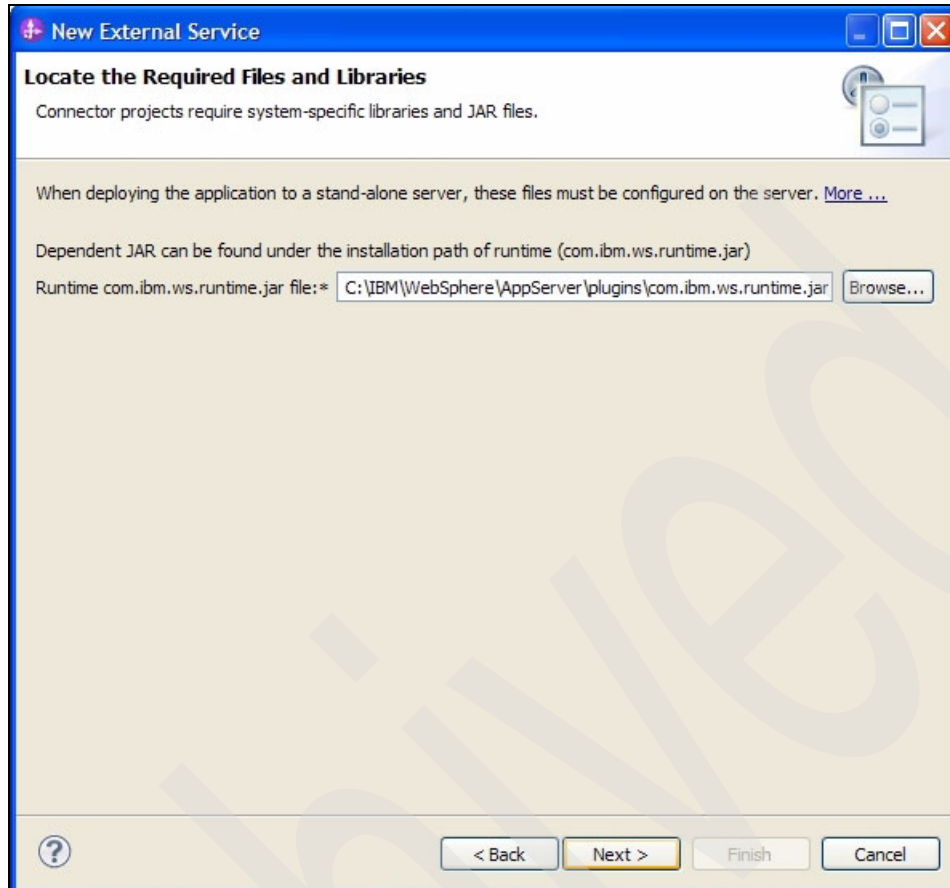


Figure 3-23 Select the runtime dependency file from the local WebSphere ESB installation directory

11. Provide the connection-related information to enable the adapter to connect to the target FileNet system (Figure 3-24 on page 94).

New External Service

Specify the Discovery Properties

Connection properties

CMIS connection information

Connection: http

Server: * 9.184.166.126

Port number: * 9080 - WebSphere Default

Path: * fncmis

User name: administrator

Password: *****

Repository Service endpoint: * http://9.184.166.126:9080/fncmis/RepositoryService

Change the logging properties for the wizard

< Back Next > Finish Cancel

Figure 3-24 Enter the FileNet connection information

12. Although you need to type the FileNet server's IP address or host name, Port number, common management information service (CMIS) Path, User name, and Password, several of these fields have default values that are auto-populated. If you have not changed these values during the FileNet installation process, using the default values typically works.

Important: The FileNet connection-related information changes per installation. Check with your FileNet administrator for accurate login credentials and other details. Refer to the IBM WebSphere Adapter for Enterprise Content Management product documentation for more details about each of these properties:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r5mx/index.jsp?topic=/com.ibm.wsadapters.jca.ecm.doc/doc/cbp_ecm_newinthisrelease.html

13. After entering the valid values for the FileNet connection-related properties, click **Next**. This action displays a wizard page and tries to connect to FileNet to fetch the object store and its children (Figure 3-25 on page 95).

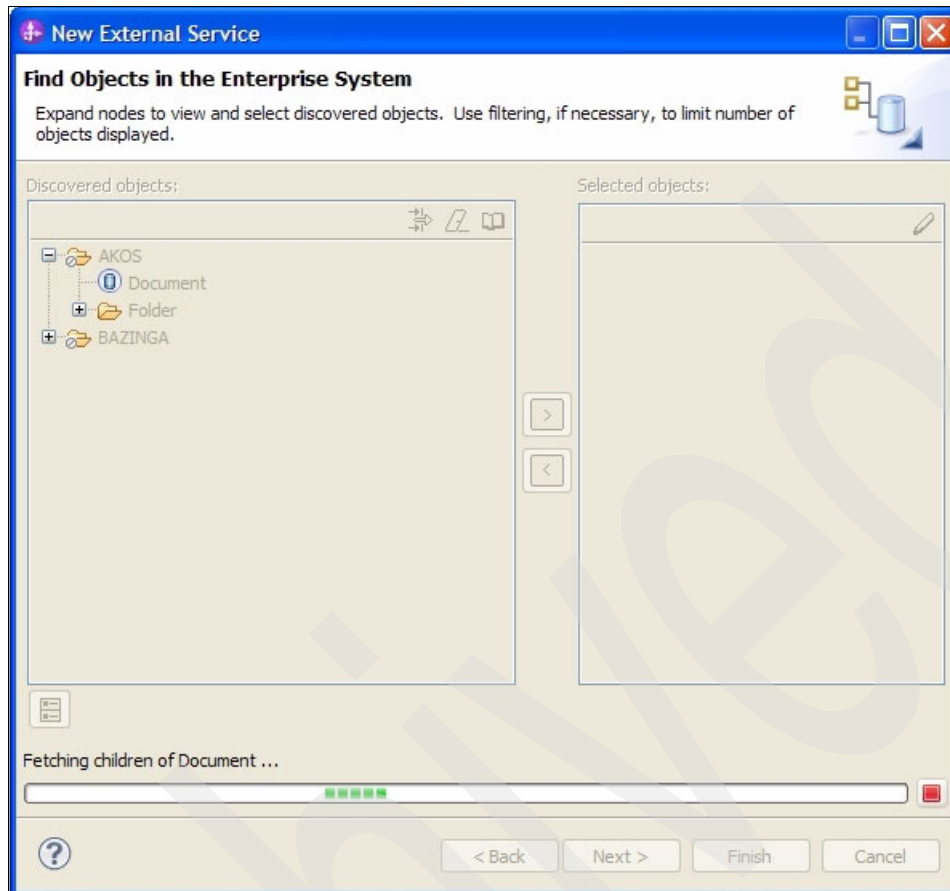


Figure 3-25 Adapter discovery process connecting to the FileNet system

14. After the wizard successfully fetches all the content from FileNet, the wizard page renders the content in a tree-like structure, as shown in Figure 3-26 on page 96.

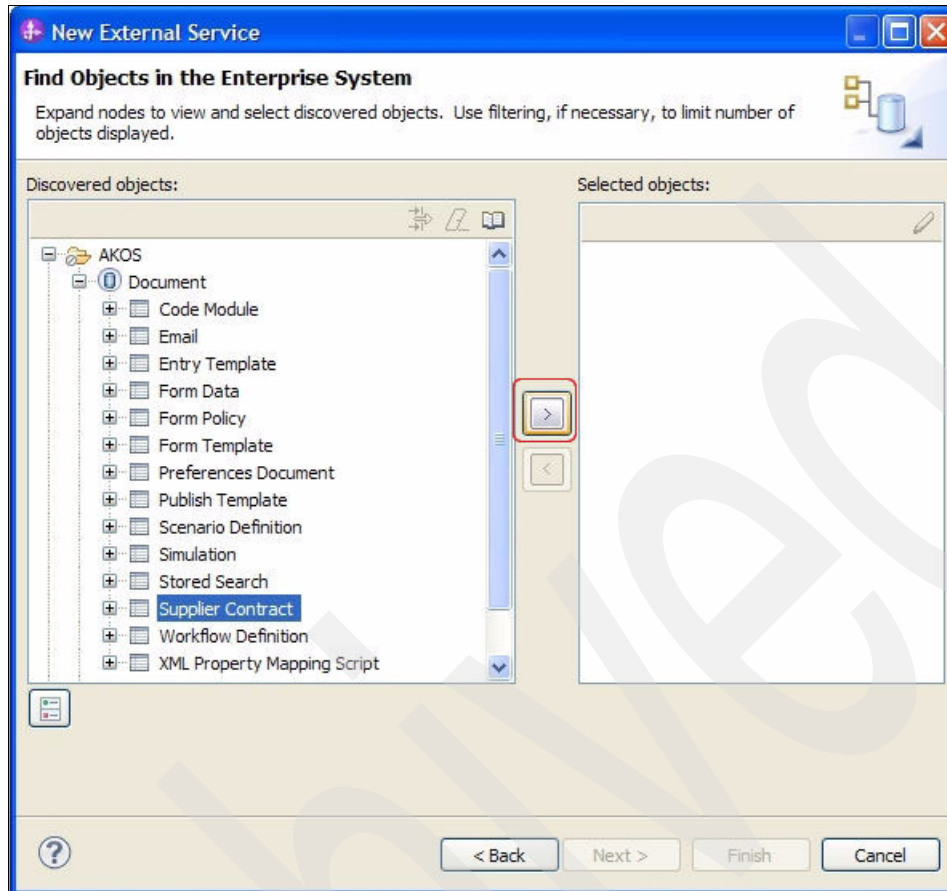


Figure 3-26 Adapter displays the FileNet content in a metadata tree structure

15. Select the **Supplier Contract** document class that you created, which was explained in section 3.4.1, “Configuring FileNet with custom Document Class type to manage legal contract documents electronically” on page 78. Move it to the Selected objects list by clicking the highlighted > icon.

This action opens the wizard to allow you to configure the properties that are associated with the selected document class (Figure 3-27 on page 97).

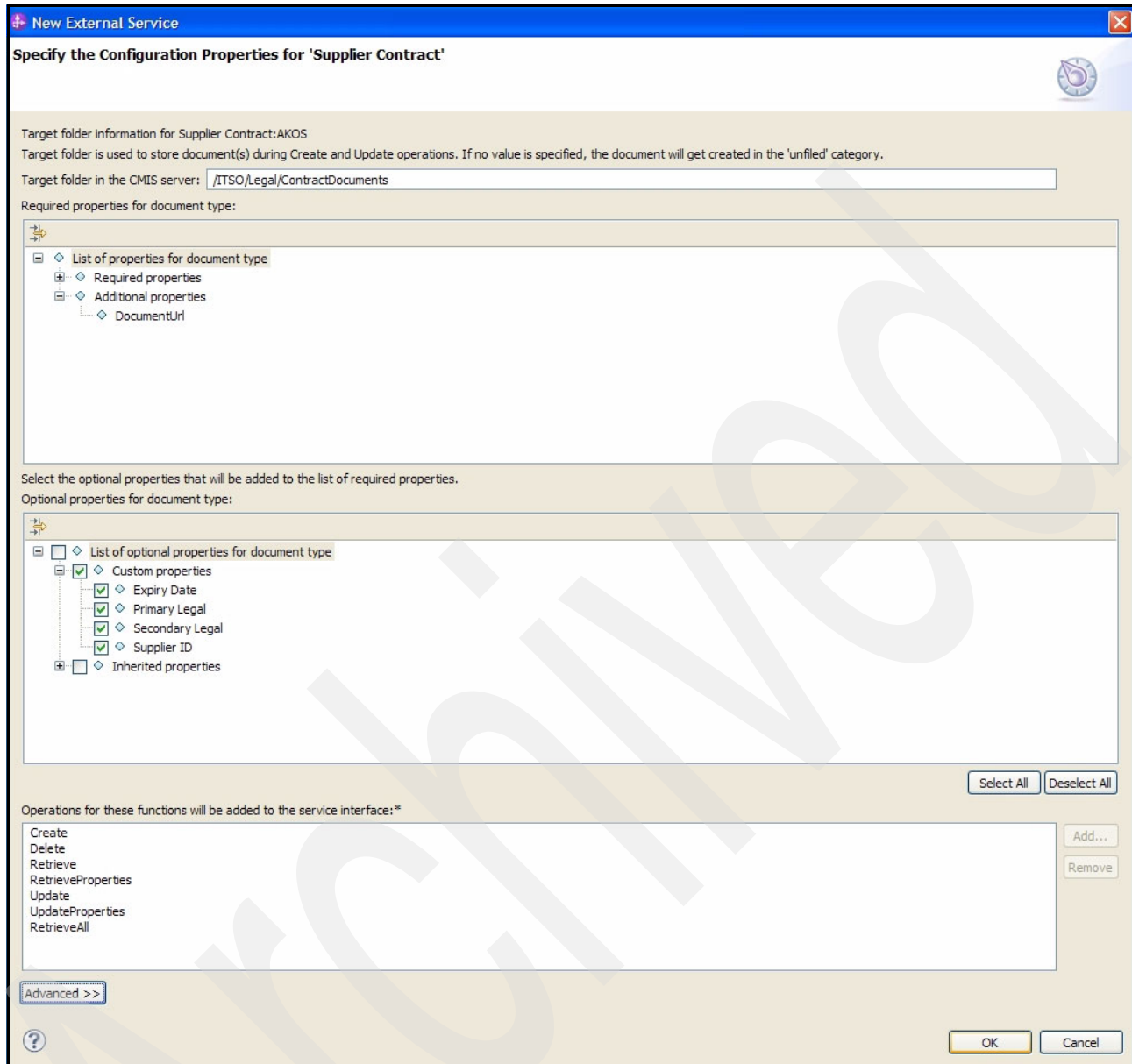


Figure 3-27 Customize the document class properties for the selected object

Look at these configurations in detail:

- ▶ **Target folder in the CMIS server:** The name of the target folder in FileNet server where the newly created document will be stored.
- ▶ **Required properties for document type:** The mandatory properties that are added to the business object that was generated for the selected document class. You cannot deselect these properties.
- ▶ **Optional properties for document type:** Select the optional properties that you want in your final business object from the list of Custom properties and Inherited properties. We added the custom properties to the document class at the time of its creation. Select all the properties from the Custom properties list. Do not select any properties from the list of inherited properties.

- ▶ **Service interface operations:** The control box on the bottom of the page allows you to customize the operations to be created for each document class that was selected. Select the default list of all operations.

For more information: See the “Selecting and Configuring business objects” section of the IBM WebSphere Adapter for Enterprise Content Management product documentation for more details about customizing the business object generation with the help of specific properties for the select document class:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r5mx/topic/com.ibm.wsadapters.jca.ecm.doc/doc/stecm_outconfig_objects.html

16. Click **OK**. You have completed the wizard page, and the selected document class is moved to the Selected objects list on the right (Figure 3-28).

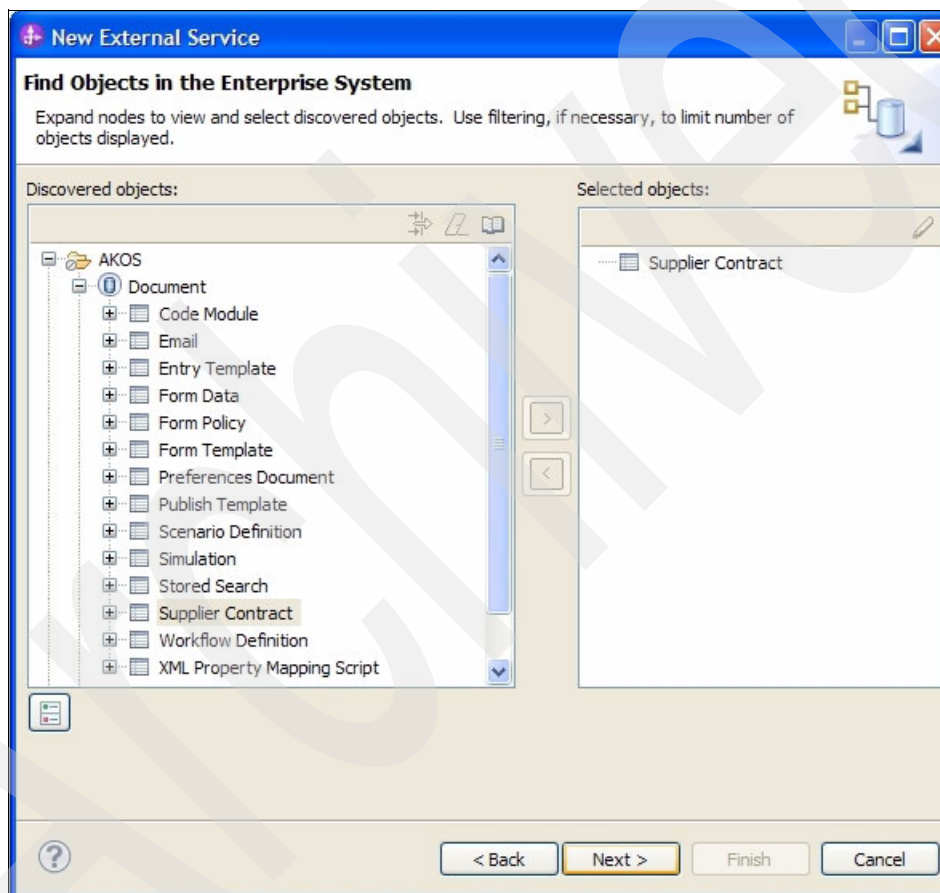


Figure 3-28 Select document class is added successfully to the Selected objects list panel

17. Next, configure the relative path of the folder in which to store the generated business objects. Keep the default value and click **Next**. See Figure 3-29 on page 99.

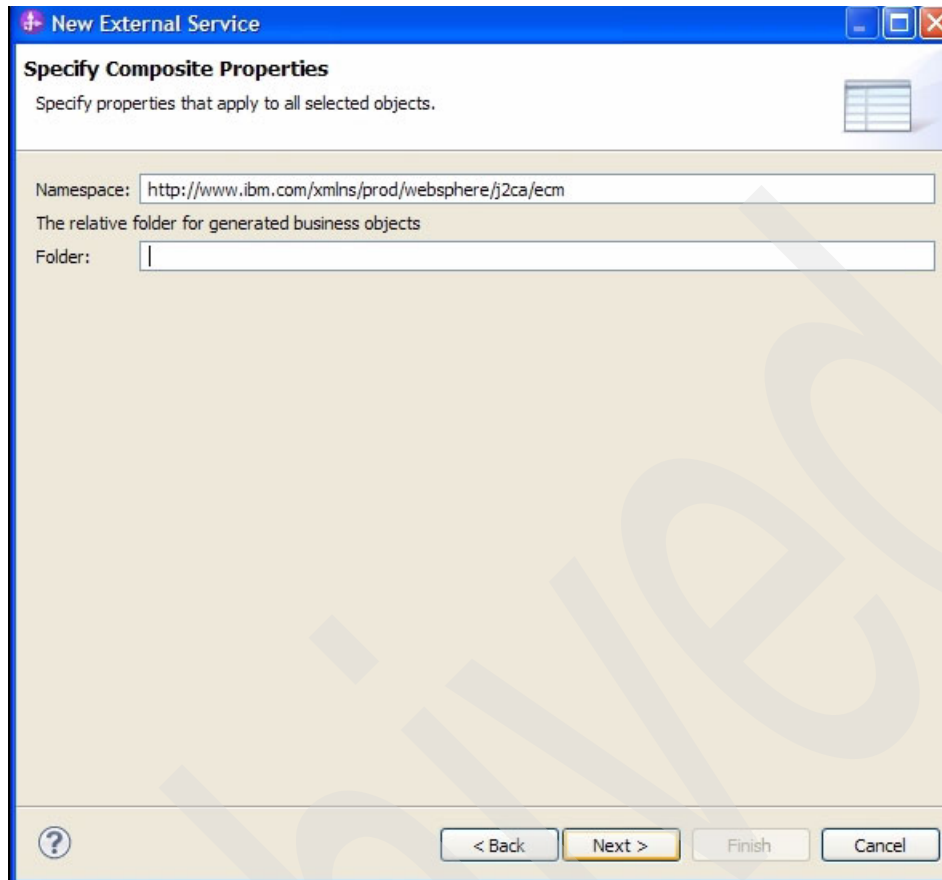


Figure 3-29 Target relative folder name to store generated business objects

18. Next, you configure the service generation and the deployment properties that are needed while deploying the adapter on the runtime WebSphere Enterprise Service Bus. See Figure 3-30 on page 100.

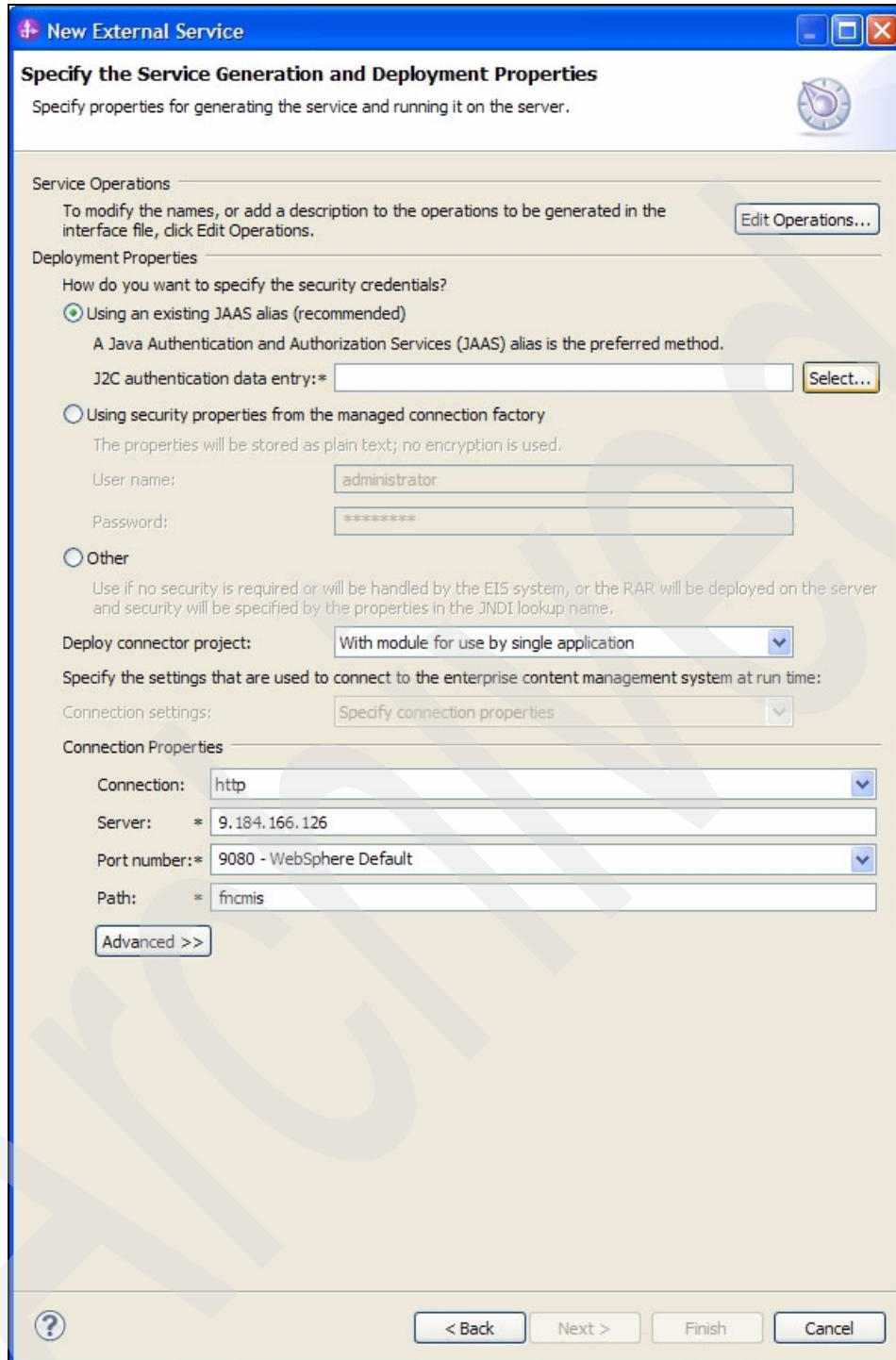


Figure 3-30 Configure service generation and deployment properties

19. In the Deployment Properties section, click **Using an existing JAAS alias**, which is the IBM-recommended approach to specifying the required security credentials for the adapter to connect to the target FileNet while deployed in WebSphere Enterprise Service Bus. Click **Select**. This action opens a new wizard page (Figure 3-31 on page 101).

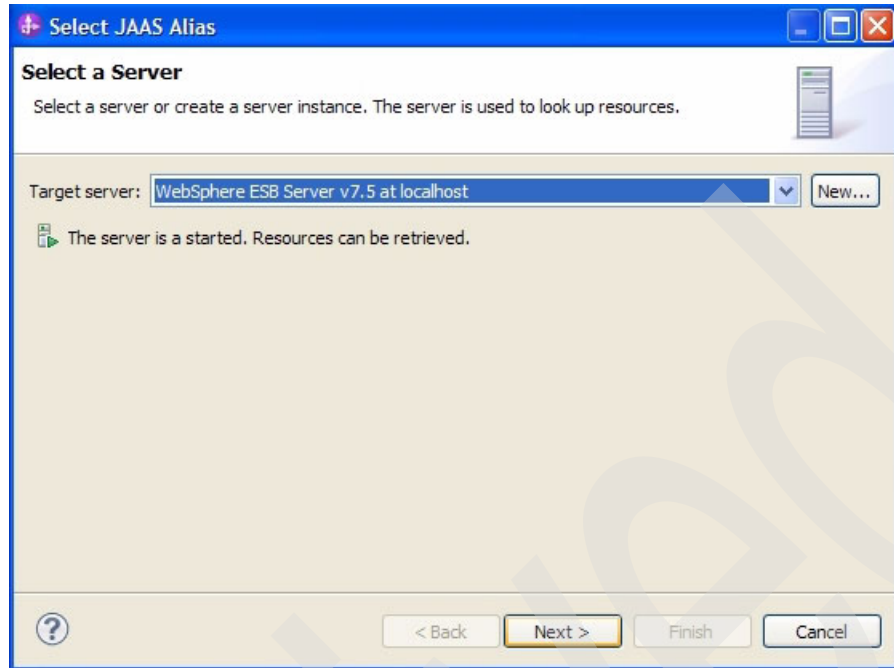


Figure 3-31 Choose the target server to configure the JAAS alias

20. Select the Target server name from the drop-down list where you want to deploy the final application and click **Next**.
21. Choose the correct JAAS alias that has been configured to connect to the target FileNet system and click **Finish**. See Figure 3-32.

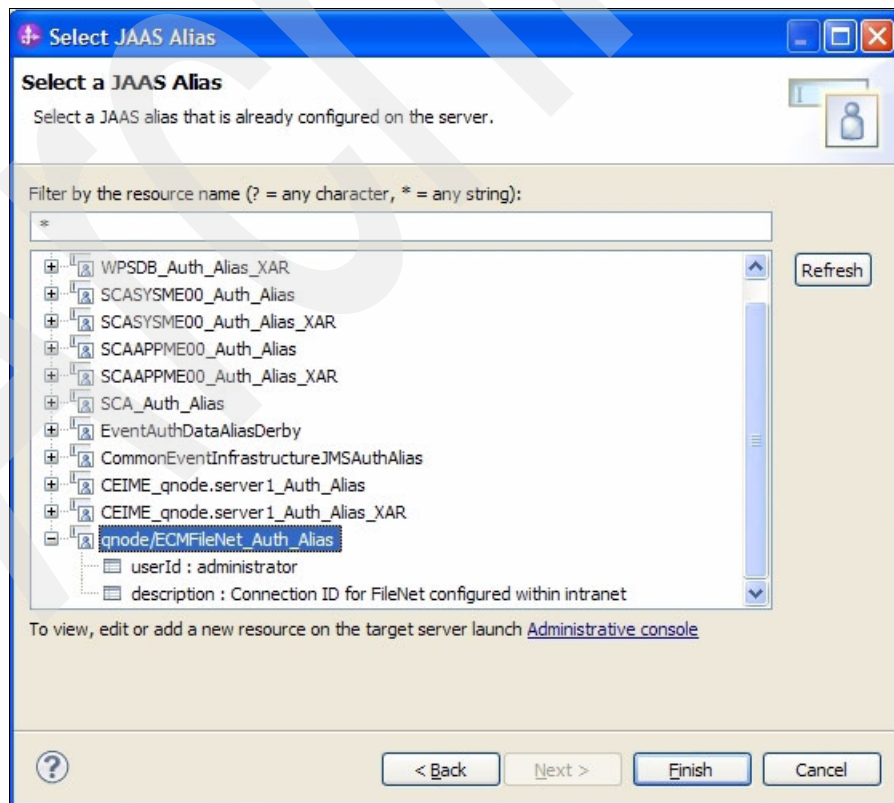


Figure 3-32 Choose the JAAS alias to connect to FileNet

For more information: See the WebSphere Enterprise Service Bus product documentation for the instructions to configure the JAAS alias using the target server's administrative console. Also, check with FileNet's administrator for the login credentials to use while setting up the JAAS alias.

22. On the next page click **Finish** to complete the external service process. See Figure 3-33.

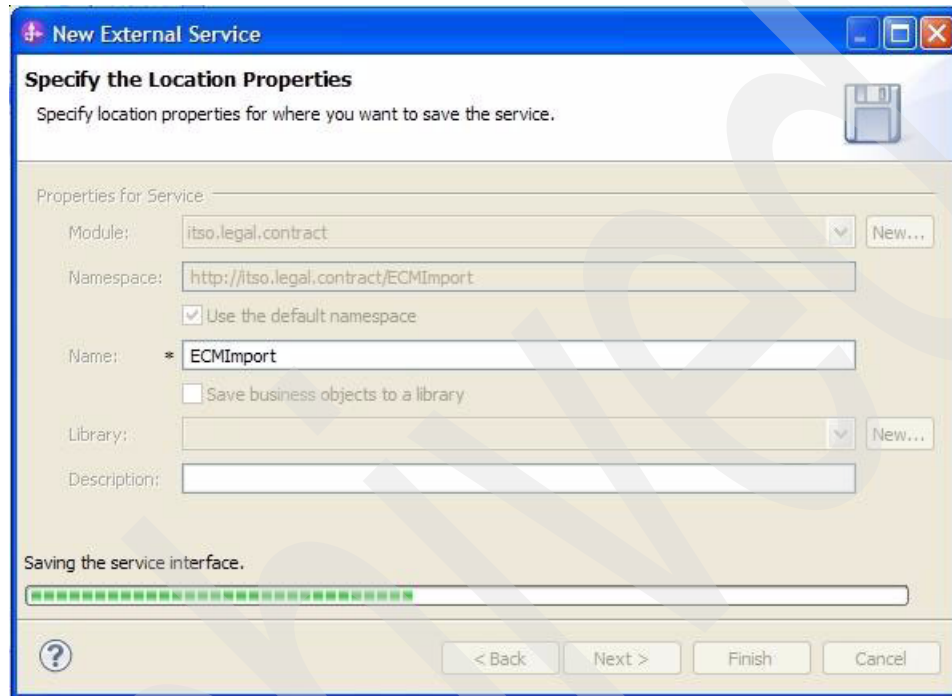


Figure 3-33 External service generating required artifacts

Using the built-in test client to quickly validate the services

It is always a good practice to test the individual components before assembling them into a bigger module. IBM WebSphere Enterprise Service Bus provides excellent capabilities to quickly validate each component that is being developed in an independent fashion. Now, we look at how to validate the services that we created using IBM WebSphere Adapter for Enterprise Content Management. Follow these steps:

1. Name the service FileNetServices. See Figure 3-34 on page 103.

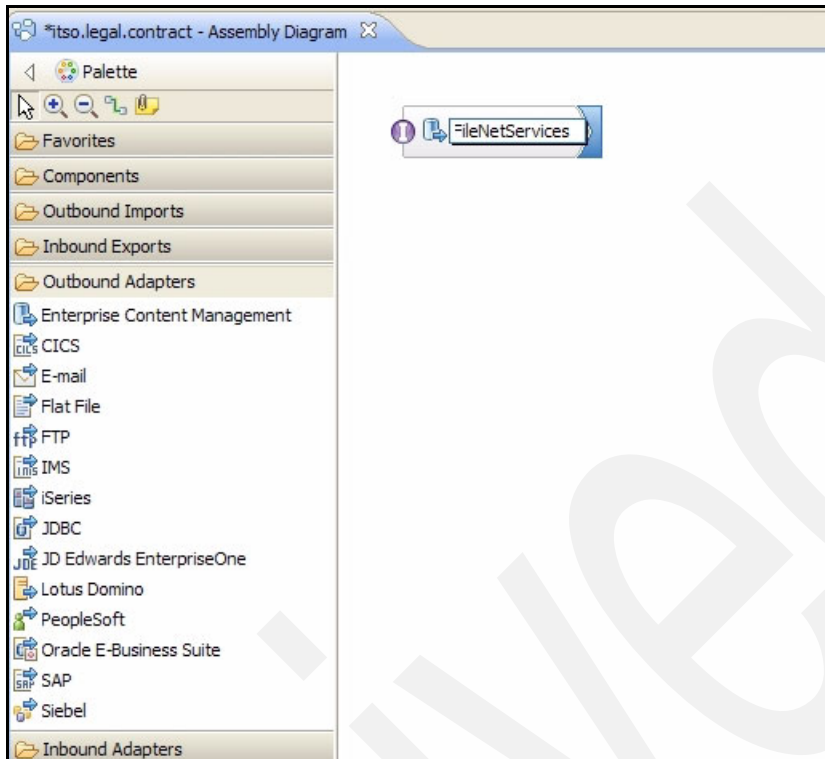


Figure 3-34 Name the service FileNetServices

2. Right-click the server and choose **Add and Remove** to deploy the component that we created on the server. See Figure 3-35 on page 104.

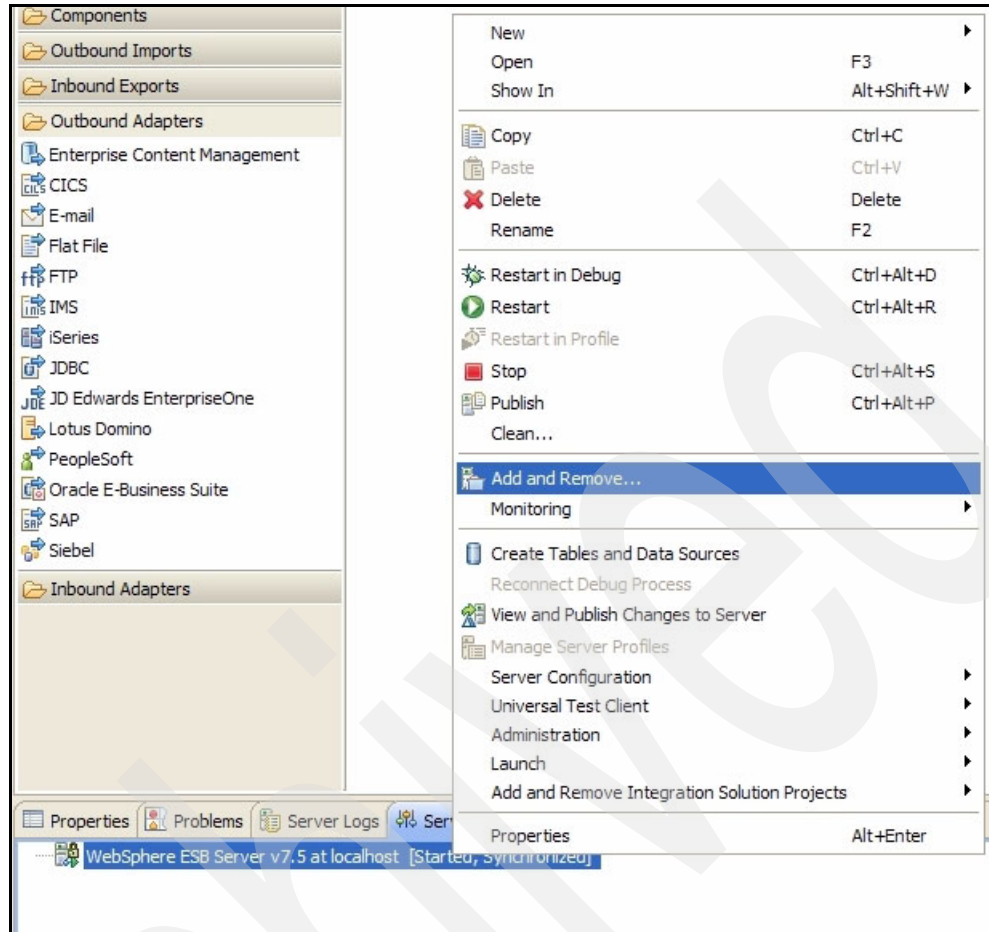


Figure 3-35 Add the project to the server for deployment

3. On the next wizard page, choose the project **its0.legal.contractApp** and click **Add >**, as shown in Figure 3-36 on page 105.

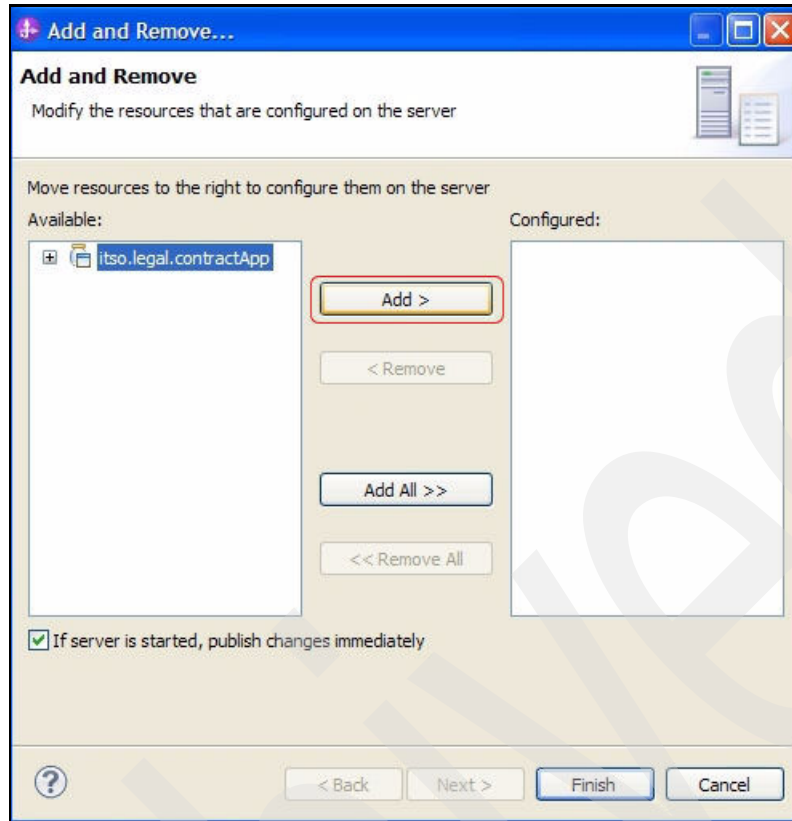


Figure 3-36 Prepare the project for deployment

4. After you move the project to the Configured section on the right side, click **Finish**. You have completed the project deployment. See Figure 3-37 on page 106.

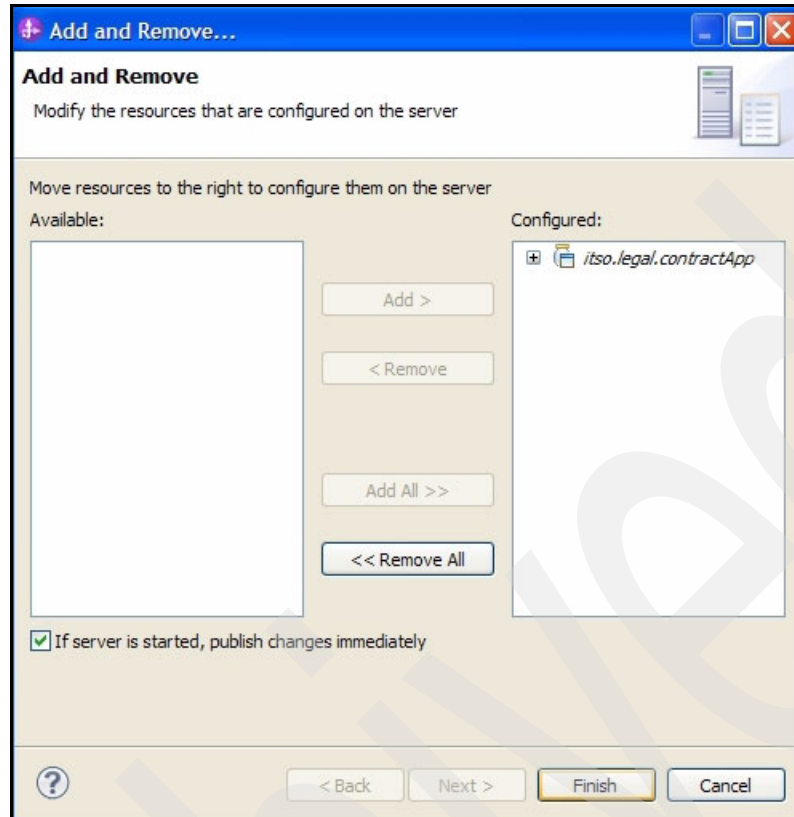


Figure 3-37 Click Finish to complete the project deployment process

5. The amount of time that is required to deploy the project on the server depends on the size of the project and the machine's hardware capabilities. Wait for few minutes for the server to complete the deployment process. After the project is successfully deployed on the server, the project's name appears under the running server with its status showing as [Started, Synchronized] (Figure 3-38).

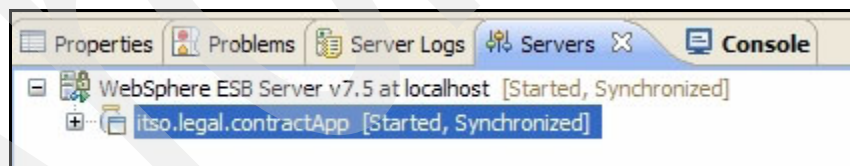


Figure 3-38 Project successfully deployed to server

6. Now that the project has been successfully deployed on the server, the next step is to invoke the built-in test client to validate the services. Right-click the **FileNetServices** component on the Assembly Diagram and choose **Test Component**, as shown in Figure 3-39 on page 107.

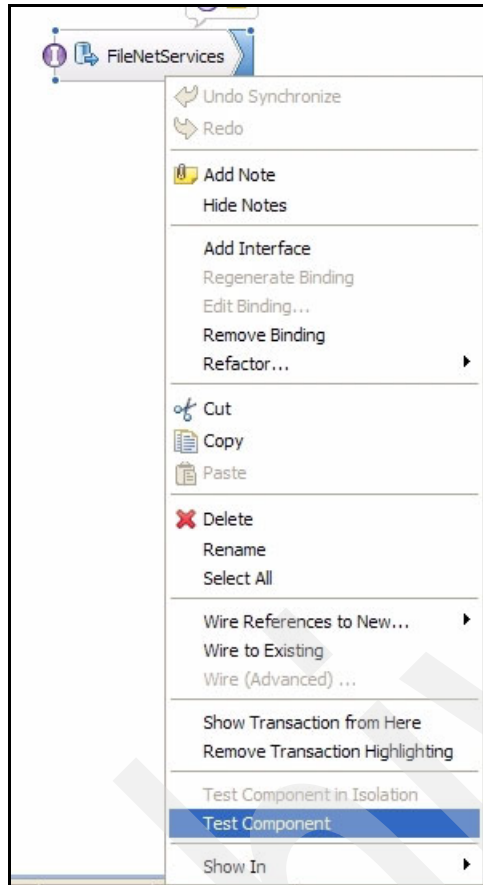


Figure 3-39 Launch the Integration Test Client component of WebSphere ESB

7. The Test Component option launches the Integration Test Client. Choose the **createAKOSSupplierContract** operation from the Detailed Properties list. Fill in the values for the required properties, as shown in Figure 3-40 on page 108.

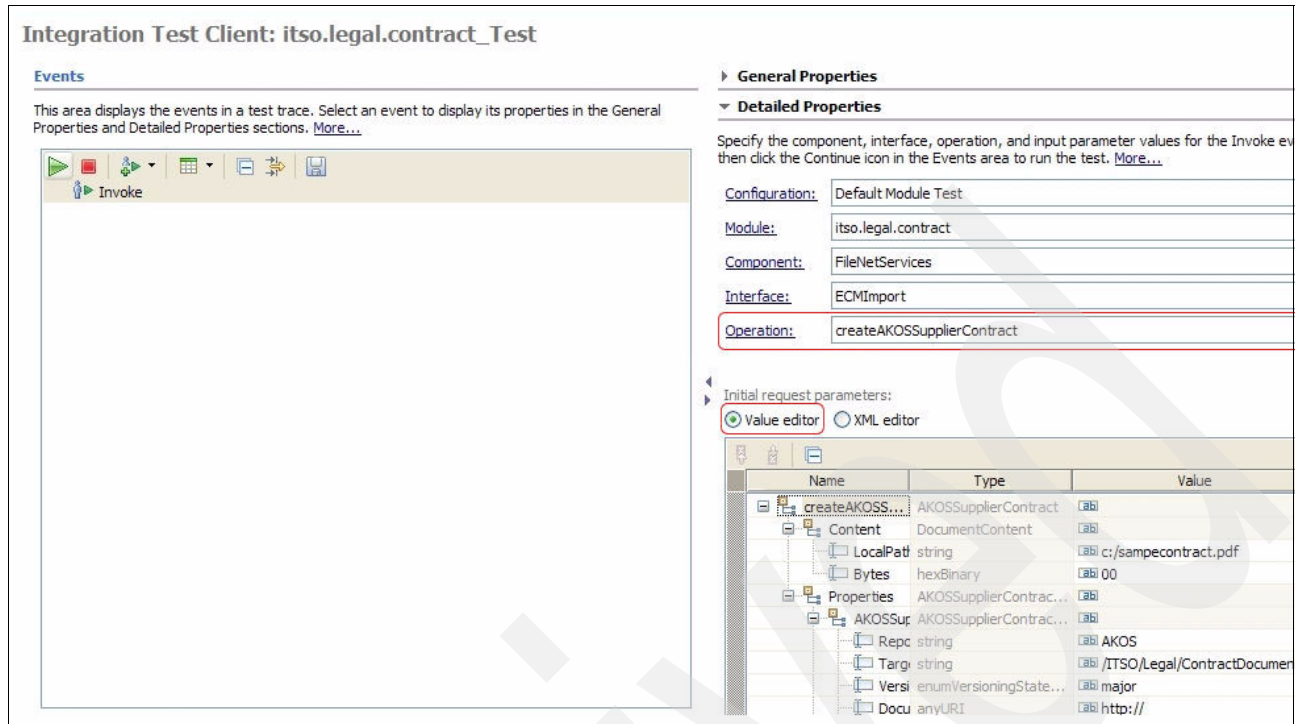


Figure 3-40 Integration Test Client to supply values for validation

- Figure 3-41 on page 109 is an expanded version of the value editor where the appropriate values are set for the required properties. Out of all the properties listed, only a few properties are mandatory for the adapter to work. We list the mandatory properties in Table 3-2.

Table 3-2 Mandatory fields

Property name	Description
LocalPath	Absolute path of the file that needs to be attached as content while creating the document in FileNet. The adapter reads the file content using the local path while processing this outbound operation.
RepositoryId	Target object store name to save the document.
TargetFolder	An optional property to specify the target location of the document. If not specified, the created document is stored under the unfiled category.
VersioningState	Versioning state of the document. The valid values are major, minor, and none.
cmisU58name	Title of the target document.
ExpiryDate	Expiry date of the supplier contract document.
PrimaryLegal	Primary legal officer's name.
SecondaryLegal	Name of secondary legal officer.
SupplierID	Unique identification number of the supplier under which the contract documents need to be stored. The supplier ID comes from the central supplier management system.

Name	Type	
createAKOSSupplierContractInput	AKOSSupplierContract	[tab]
Content	DocumentContent	[tab]
LocalPath	string	[tab] c:/sampecontract.pdf
Bytes	hexBinary	[tab] 00
Properties	AKOSSupplierContractDocumentProperties	[tab]
AKOSSupplierContractCommonProperties	AKOSSupplierContractCommonProperties	[tab]
RepositoryId *	string	[tab] AKOS
TargetFolder	string	[tab] /TSO/Legal/ContractDocuments
VersioningState	enumVersioningState <string>	[tab] major
DocumentUrl	anyURI	[tab] http://
PathSegment	string	[tab]
MimeType	string	[tab]
RetrieveCriteria	AKOSSupplierContractRetrieveCriteria	[tab]
SelectClause	string	[tab] *
WhereClause	string	[tab]
AKOSSupplierContractProperties	AKOSSupplierContractProperties	[tab]
IsInExceptionState *	boolean	[tab] false
IsReserved *	boolean	[tab] false
IsVersioningEnabled *	boolean	[tab] false
MajorVersionNumber *	integer	[tab] 0
MinorVersionNumber *	integer	[tab] 0
VersionStatus *	integer	[tab] 0
cmisU58name	string	[tab] samplecontract_1
cmisU58objectId	string	[tab]
cmisU58objectTypeId *	string	[tab] SupplierContract
ExpiryDate	dateTime	[tab] 2011-07-10T17:13:49.640+0530
PrimaryLegal	string	[tab] Rajan Kumar
SecondaryLegal	string	[tab] Peter Broadhurst
SupplierID	string	[tab] MTS6548902

Figure 3-41 Integration Test Client populated with test values

- After you enter all the required properties, click the green right arrow icon to start the validation process. We highlighted the green arrow icon in Figure 3-42.



Figure 3-42 Invoke the validation process

- When the Integration Test Client is used to test the project for the first time, it is a requirement to attach the project to a target server, as shown in Figure 3-43 on page 110.

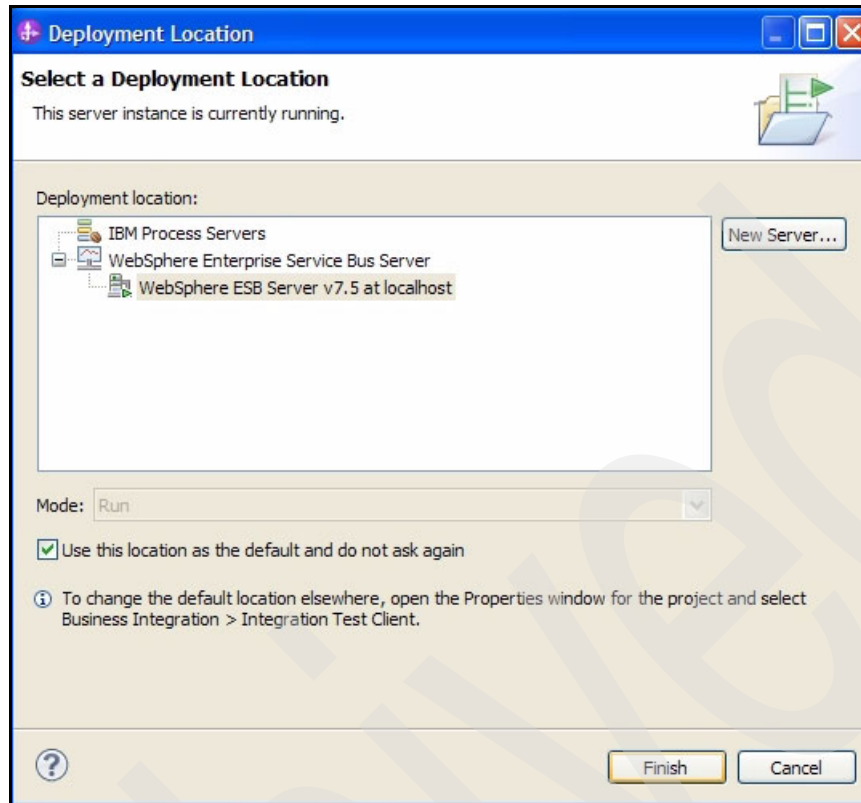


Figure 3-43 Choose the deployment location for testing the component

11. Click **Finish**, which invokes the service, as shown in Figure 3-44.

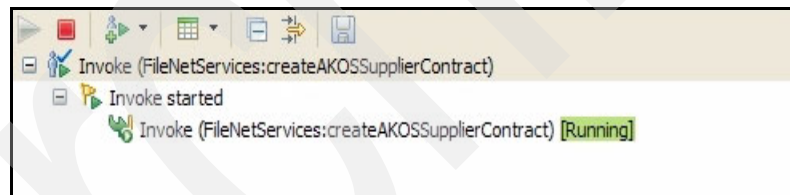


Figure 3-44 Integration Test Client running test

12. After the service is successfully invoked and executed at the FileNet end, the response from the adapter is displayed on the same editor. See Figure 3-45 on page 111.

Integration Test Client: itso.legal.contract_Test

Events

This area displays the events in a test trace. Select an event to display its properties in the General Properties and Detailed Properties sections.
[More...](#)

- Invoke (FileNetServices:createAKOSSupplierContract)
 - Invoke started
 - Invoke (FileNetServices:createAKOSSupplierContract)
 - Return (FileNetServices:createAKOSSupplierContract)
 - Invoke returned

General Properties

Module: [itso.legal.contract](#)
 Component: [FileNetServices](#)
 Interface: [ECMImport](#)
 Operation: [createAKOSSupplierContract](#)

Return parameters:

Value Editor XML Source

Name	Type	Value
createAKOSSupplierContractOutput...	AKOSSupplierCon...	
AKOSSupplierContractCommonProp...	AKOSSupplierCon...	
RepositoryId *	string	AKOS
TargetFolder	string	/ITSO/Legal/ContractDocuments
VersioningState	enumVersioningSt...	major
DocumentUrl	anyURI	http://9.184.166.126:9080/fncmis/reso
PathSegment	string	samplecontract_1
MimeType	string	
RetrieveCriteria	AKOSSupplierCon...	
AKOSSupplierContractProperties	AKOSSupplierCon...	
IsInExceptionState *	boolean	false
IsReserved *	boolean	false
IsVersioningEnabled *	boolean	true
MajorVersionNumber *	integer	1
MinorVersionNumber *	integer	0
VersionStatus *	integer	1
cmisU58name	string	samplecontract_1
cmisU58objectId	string	idd_7FBF67C1-82F5-462E-89CC-59E37
cmisU58objectTypeId *	string	SupplierContract
ExpiryDate	dateTime	2011-07-10T17:13:49.640+0000
PrimaryLegal	string	Rajan Kumar
SecondaryLegal	string	Peter Broadhurst

Figure 3-45 The response from the service exposed via adapter is shown on the editor

- If you log on to FileNet and look in the target directory that was chosen for creating the document, you see that a file was created with the title that was provided as input (Figure 3-46 on page 112). You have completed the test process.

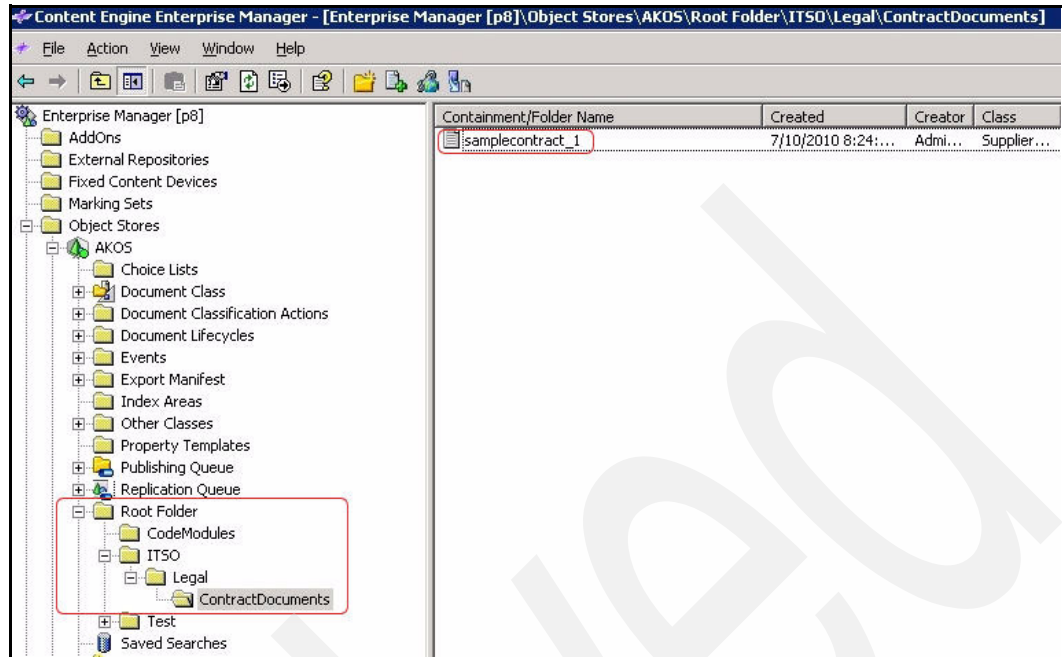


Figure 3-46 Document successfully created on FileNet

3.4.3 Building an FTP interface for the createDocument business service exposed via WebSphere Adapter

In this section, we guide you through a step-by-step process to build an application to create supplier documents on FileNet using the FTP protocol. ITSO Enterprise manages all its supplier information in its central supplier management system. The central supplier management system was developed in-house and it does not have a good interface mechanism. ITSO Enterprise decided to extend the mechanism to create supplier documents by enabling the application to create the supplier records on an FTP server along with the supplier contract information -- a local path to the scanned version of the supplier contract document. This file will be picked up by IBM WebSphere Adapter for FTP and pushed to FileNet to the create appropriate supplier contract document. Follow these steps:

1. To start, select **Data** → **New** → **Business Object** to create a new business object. See Figure 3-47 on page 113.

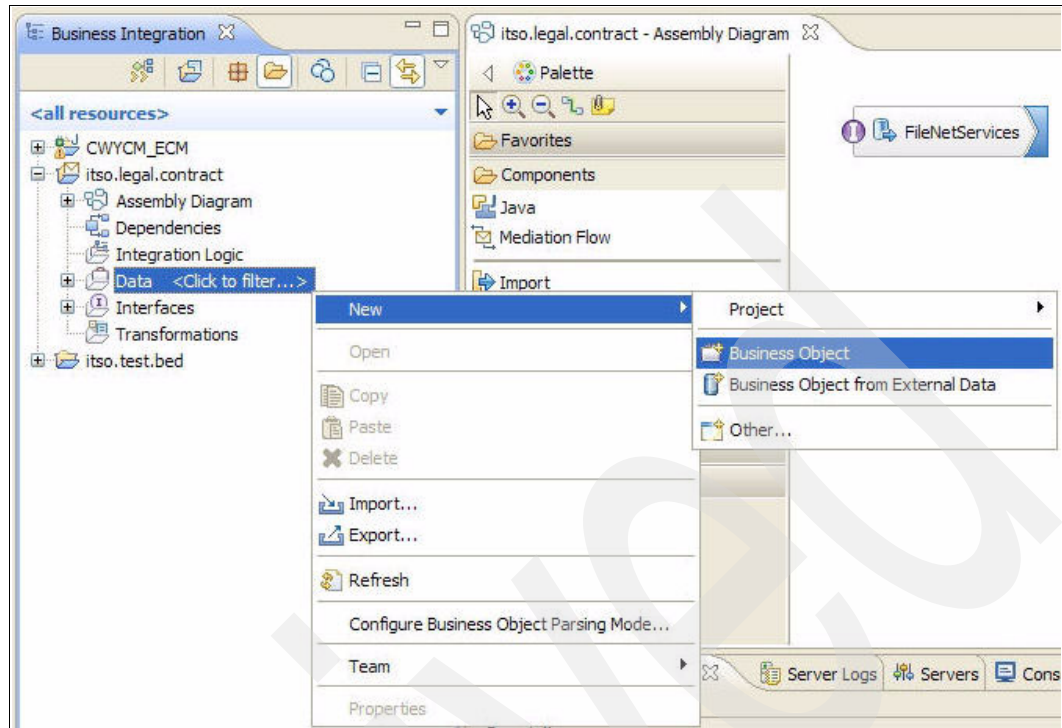


Figure 3-47 Create a new Business Object

2. The New Business Object page appears. For the Name of the business object, type `SupplierContract` and click **Finish**. See Figure 3-48.

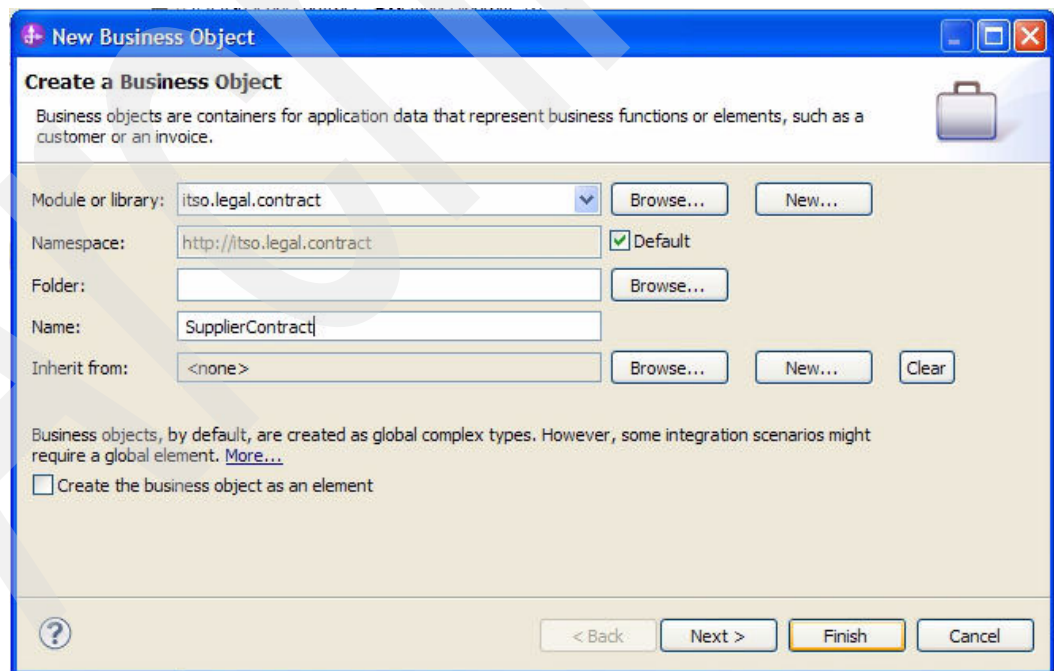


Figure 3-48 Name the business object `SupplierContract`

3. This action creates a new business object and places it on the Assembly Diagram for further edits. Right-click your new business object and choose **Add field**, as shown in Figure 3-49 on page 114.

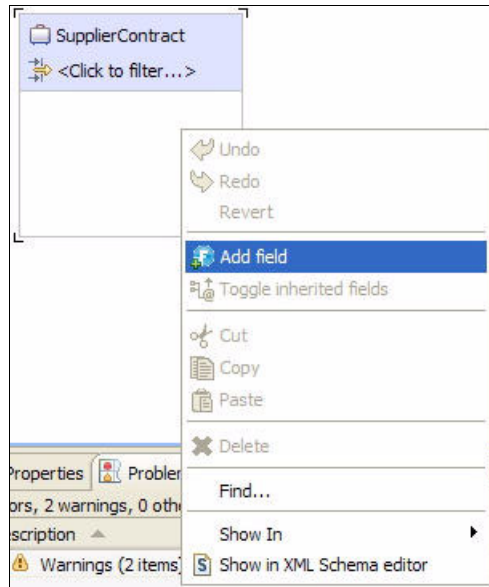


Figure 3-49 Add fields to the business object

4. For the Name of the field, type `primarylegal` and select **string** for its data type.
5. Repeat steps 3 and 4 to add a few more fields to complete the business object creation. Figure 3-50 shows a list of fields and their corresponding data types.

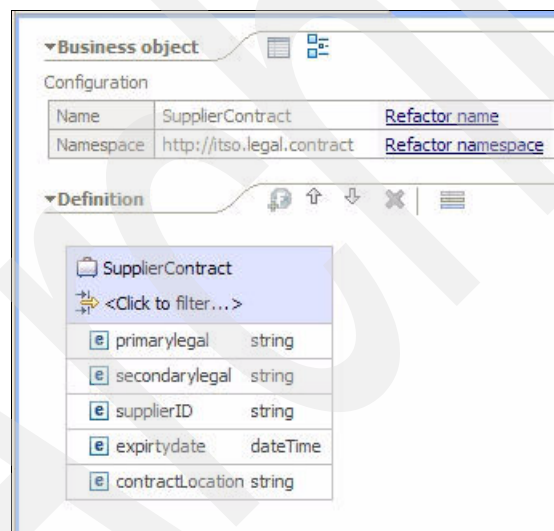


Figure 3-50 Final structure of business object

6. Start the IBM WebSphere Adapter for FTP discovery process and build the required logic to poll and parse the supplier record information arriving via the FTP channel. Drag the **FTP** control from the Inbound Adapters palette and drop it on the project's Assembly Diagram, as shown in Figure 3-51 on page 115.

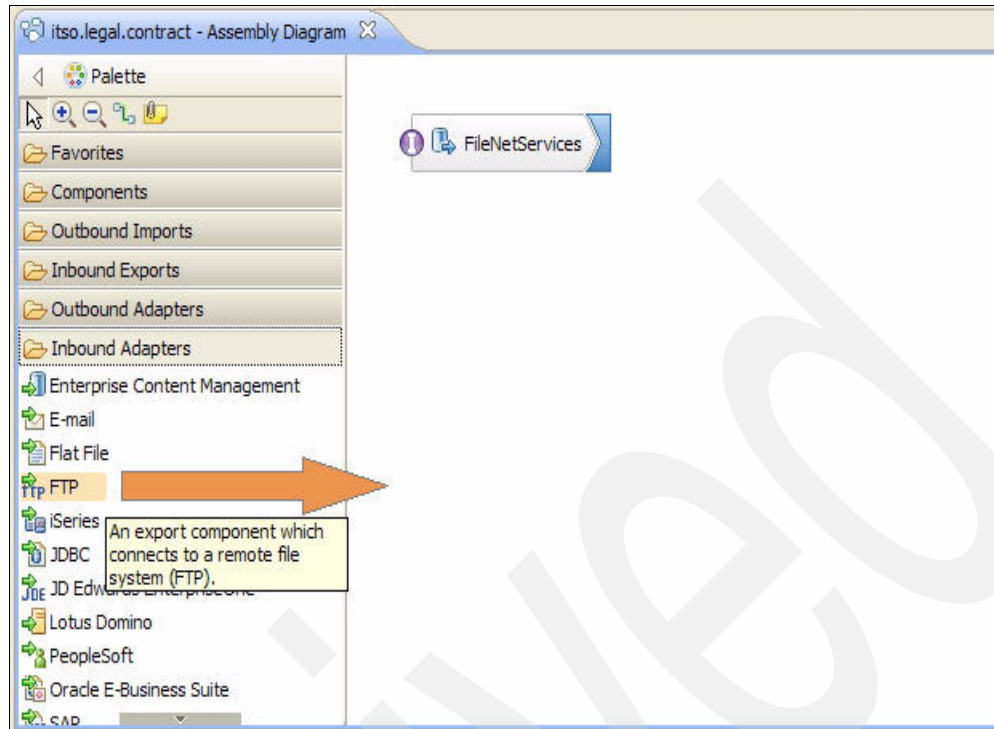


Figure 3-51 Initiate the FTP adapter's discovery process

7. On the next window, choose the **Simple: Create an inbound FTP service to read from a remote file** predefined template for faster integration with the target FTP server. Click **Next**. See Figure 3-52 on page 116.

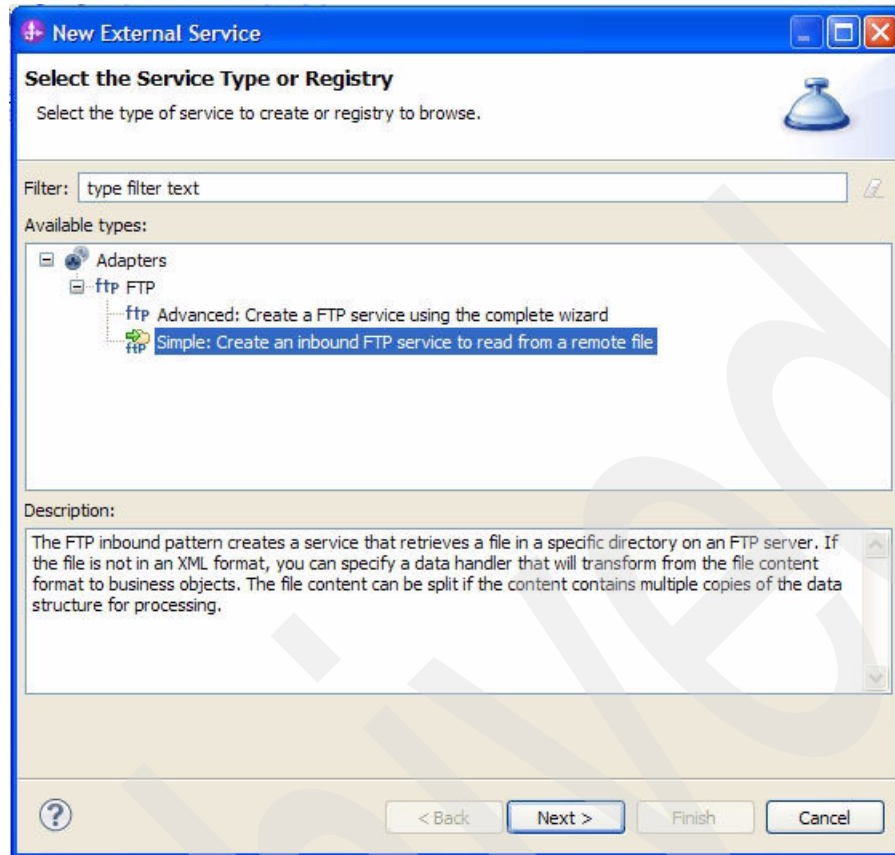


Figure 3-52 Choose predefined template approach for faster integration with FTP server

8. On the next wizard page, for the Name of the FTP service, type FTPExport and click **Next**, as shown in Figure 3-53 on page 117.

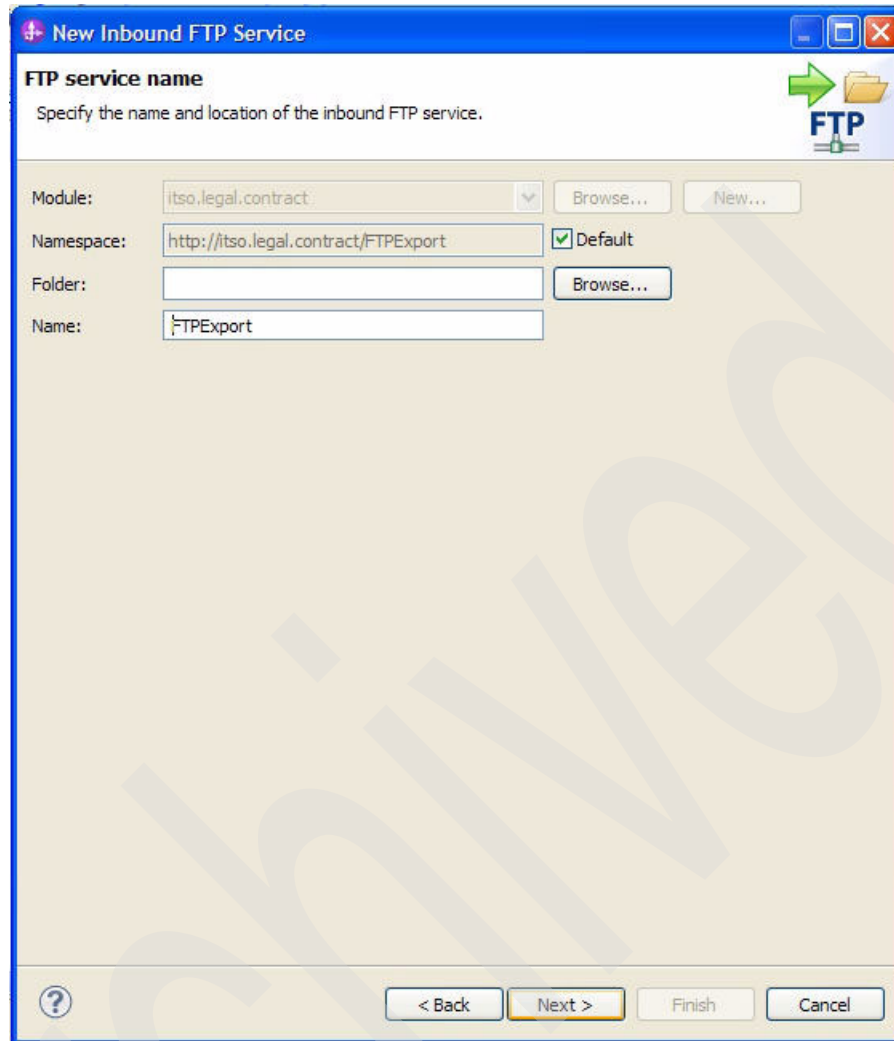


Figure 3-53 Enter the service Name FTPEXport

9. A few mandatory configuration settings are necessary for the service to work. You can see them on Figure 3-54 on page 118. For the Business Object, select **SupplierContract** (from the list of business objects, click **Browse** to find SupplierContract). For the FTP server host name, type the target FTP server host name or IP address. For the Remote directory from which the supplier contract files will be copied to the server, type the name of your source remote directory. Choose a valid Local staging directory name, because the adapter processes the files only after staging them to a local directory. Click **Next**. Figure 3-54 on page 118 has all the fields populated with appropriate values.

New Inbound FTP Service

Business object and location
Specify the business object and location for the input file.

What business object do you want to read from the input file?

Business object:

What directory should be polled for the input file?

FTP server host name:

Remote directory:

Where do you want to temporarily put the input file?

Local staging directory:

Figure 3-54 Populate the required fields for the FTP service

Your environment: Change these values to the appropriate values for your environment. Make sure that you have access to the target FTP server that you want to service-enable.

10. On the next page, configure the security credential that is needed to connect to the target FTP server. Type the user name and password required to connect to the FTP server and click **Next**. See Figure 3-55 on page 119.

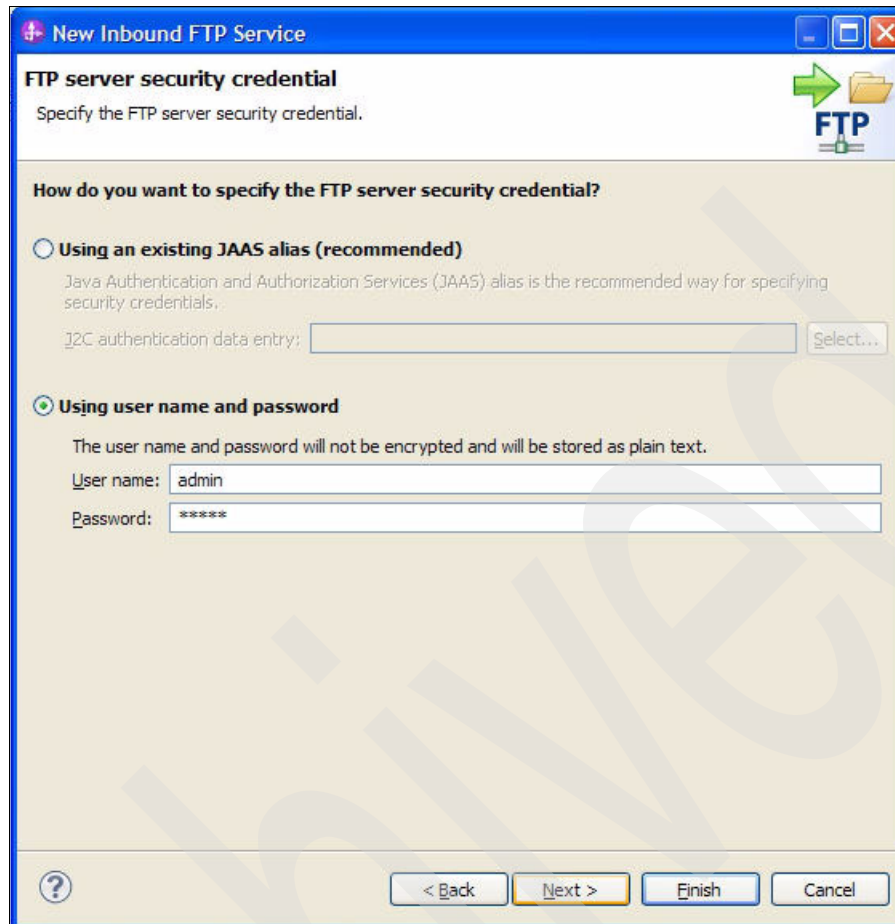


Figure 3-55 Configure the security credential for connecting to the FTP server

11. Configure the file format and its parsing logic. For the input file format, click **Other** and click **Select**. See Figure 3-56 on page 120.

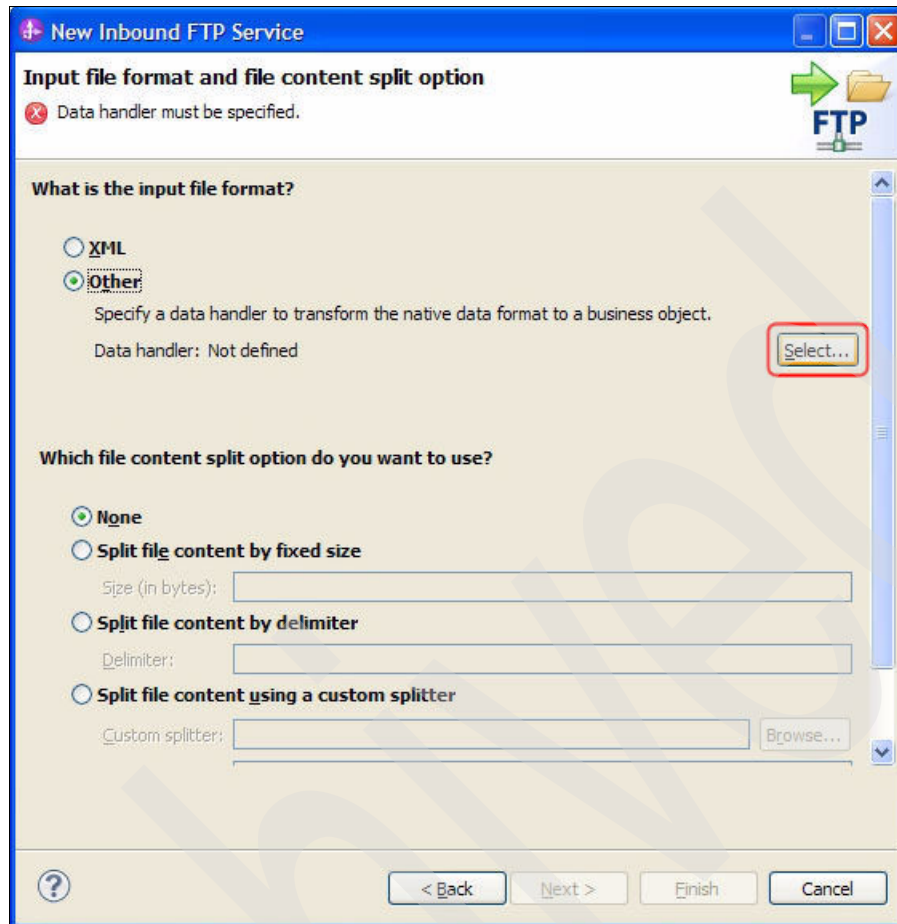


Figure 3-56 Choose the input file format

12. On Data Handler Configuration page, choose **Delimited** and click **Next**. See Figure 3-57 on page 121.

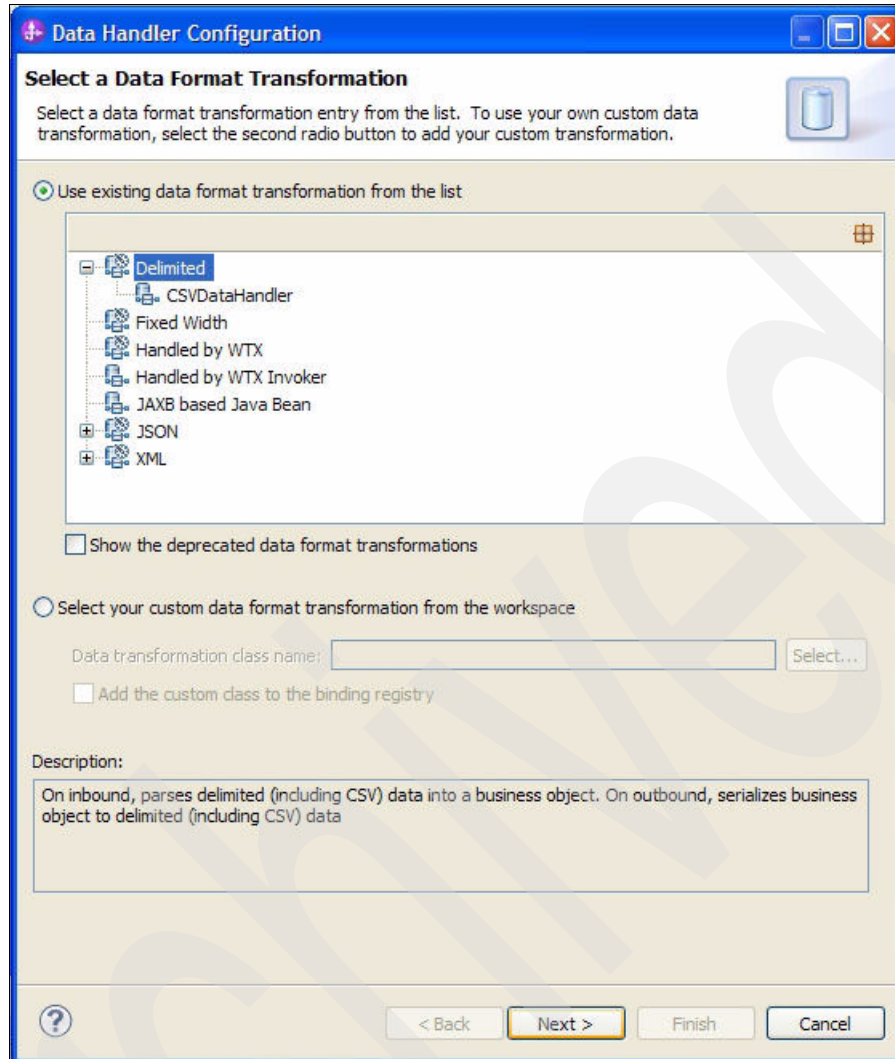


Figure 3-57 Choose the Delimited data format of the input file

13. Because the data is separated by an equal sign (=) on the input files, type the equal sign (=) for the delimiter and click **Next**. See Figure 3-58 on page 122.

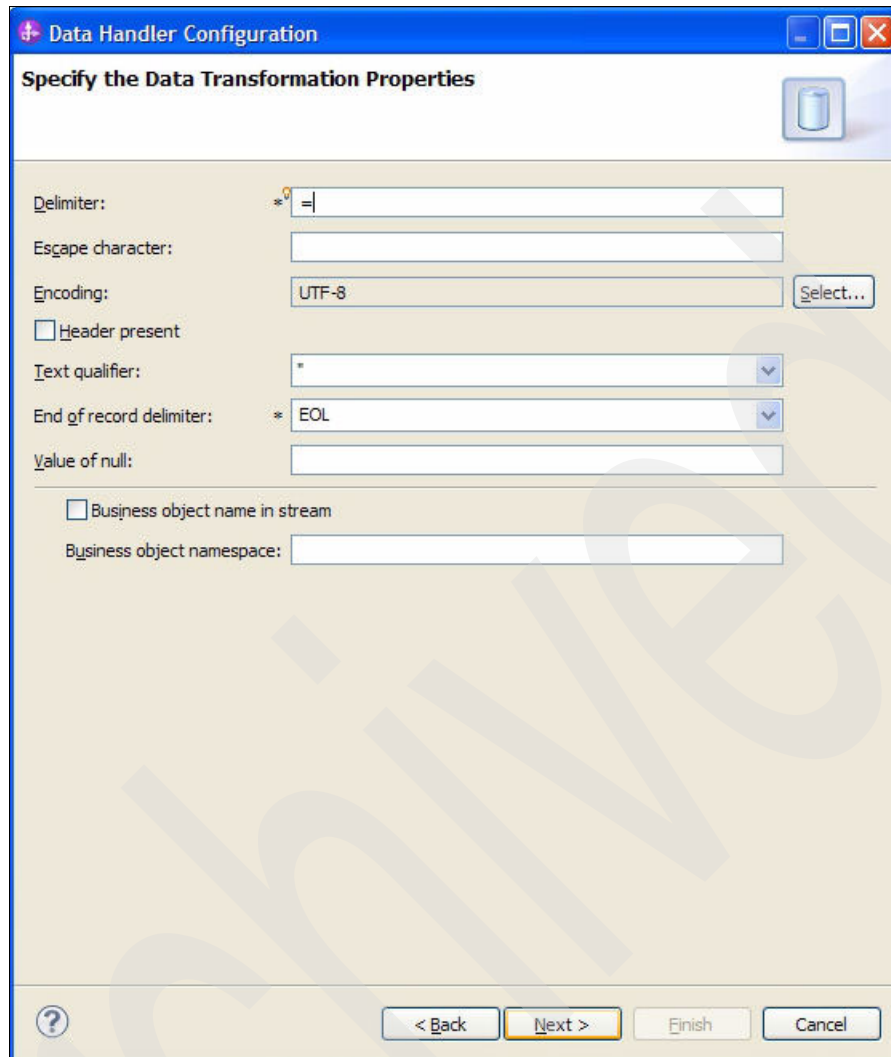


Figure 3-58 Type the Delimiter for the input file processing

14. On the next page, leave the default Name for the data transformation configuration and click **Finish**. See Figure 3-59 on page 123.

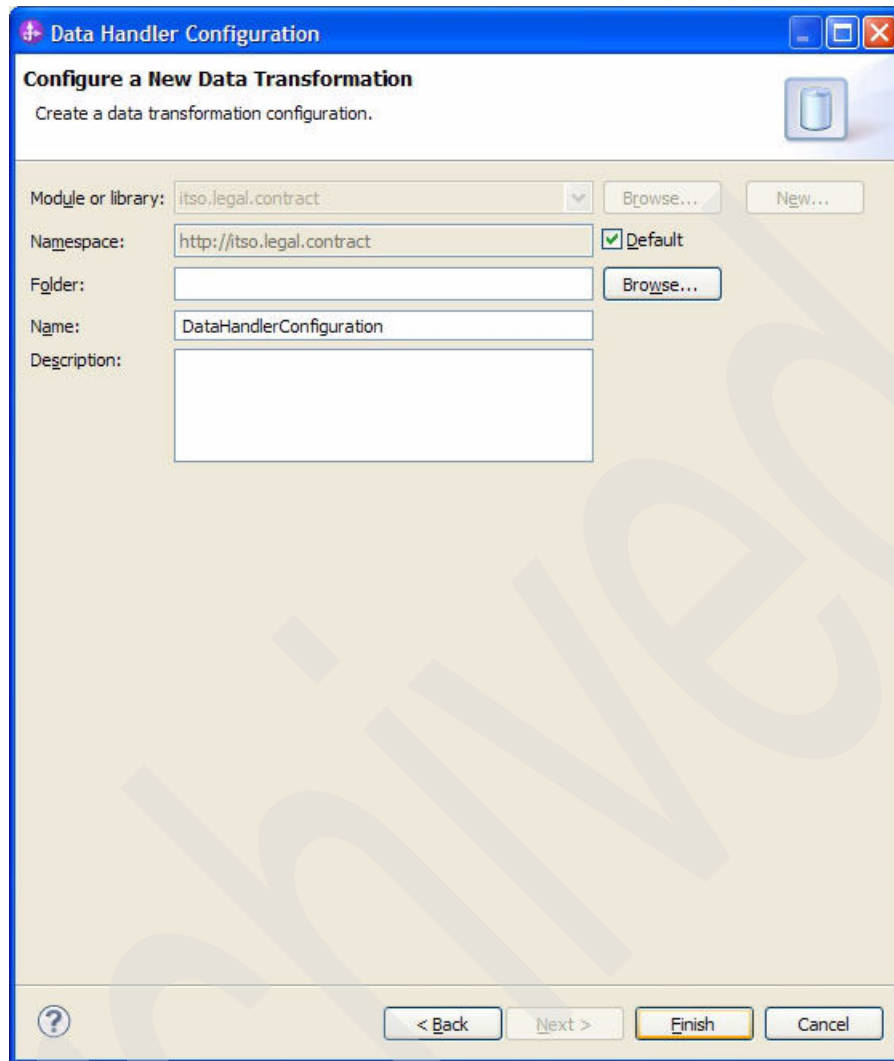


Figure 3-59 Data handler configuration

15. The wizard returns you to the same page where you started the data handler configuration. You can see the data handler name being populated with the value chosen on the previous step. Click **Finish** to complete the process. See Figure 3-60 on page 124.

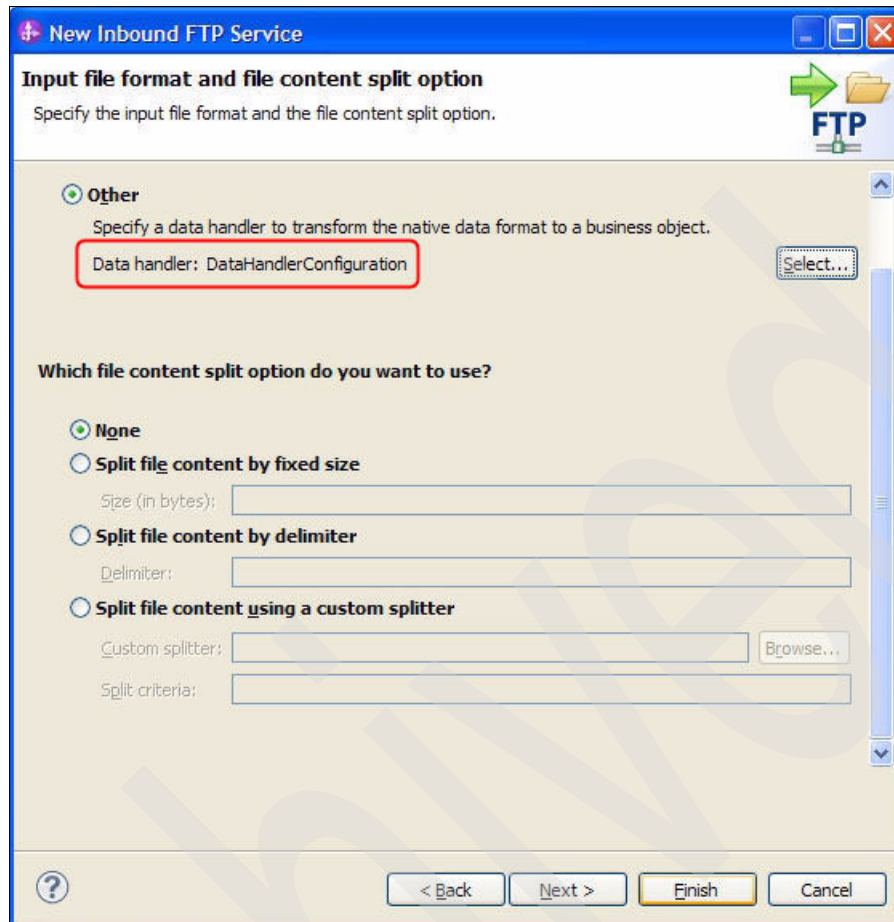


Figure 3-60 Complete the data handler configuration

16. Configuring the data handler creates another component on the Assembly Diagram for the FTPService. Rename this component ContractsOnFTP. See Figure 3-61.

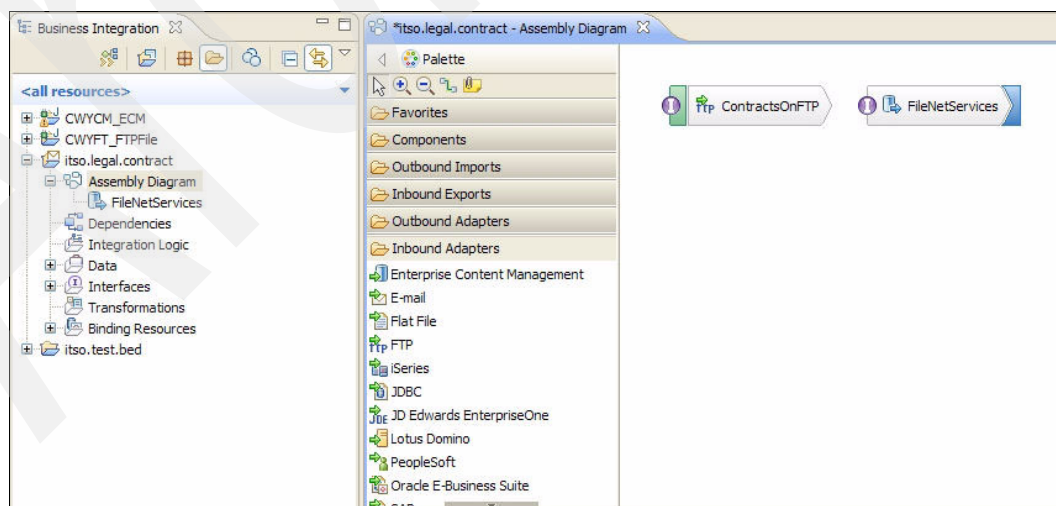


Figure 3-61 Service for FTP successfully generated and added to the Assembly Diagram

17. Build the required logic to route the information coming out of the FTP service to the FileNet service. Because these services work in separate data formats, you must create

custom mapping functionality to make the services understand each other's format. Drag and drop the **Mediation Flow** component from the Components palette on the Assembly Diagram, as shown in Figure 3-62.

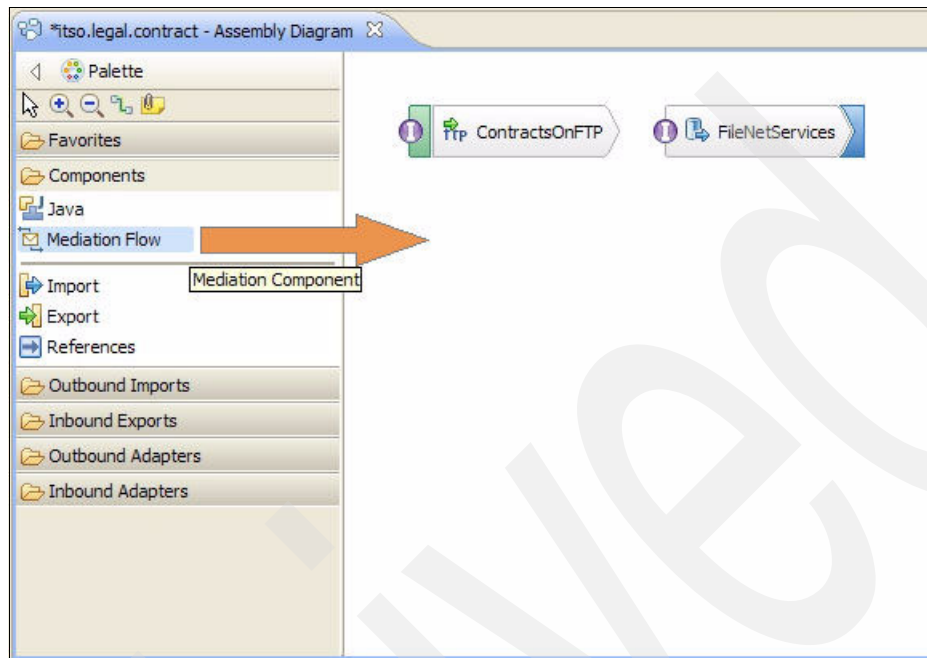


Figure 3-62 Add a Mediation Component to the project

18. Wire the FTP service **ContractsOnFTP** to the **Mediation** component. You will be asked for a confirmation to add the service interface from the export to the target. Click **OK**, as shown in Figure 3-63.

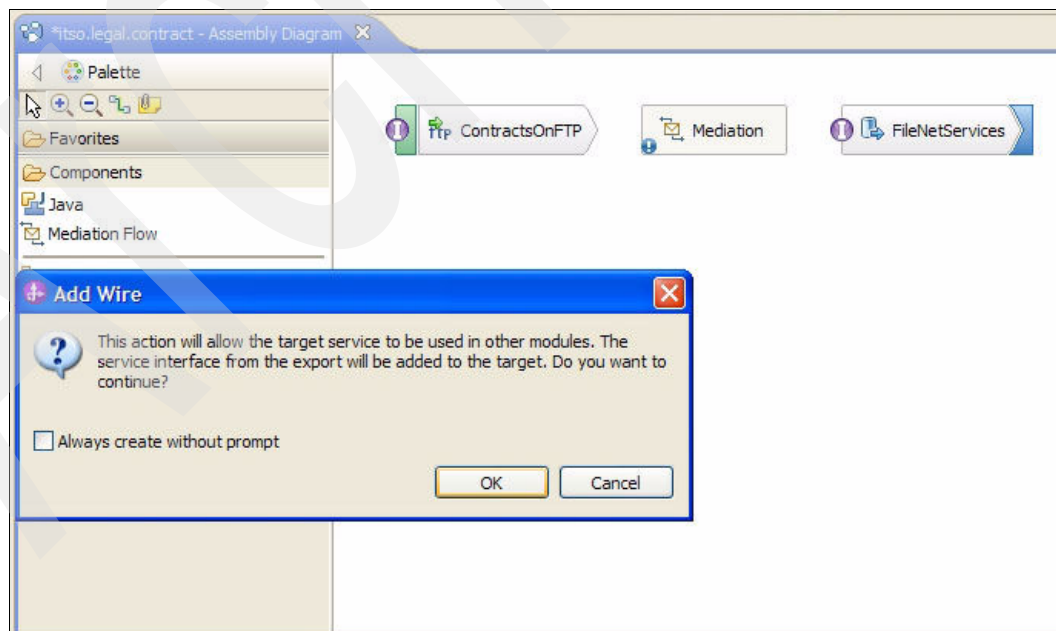


Figure 3-63 Wire the FTP service and the Mediation component

19. Wire the **Mediation** component and **FileNetServices** on the other side. See Figure 3-64. Click **OK** for the dialog box.

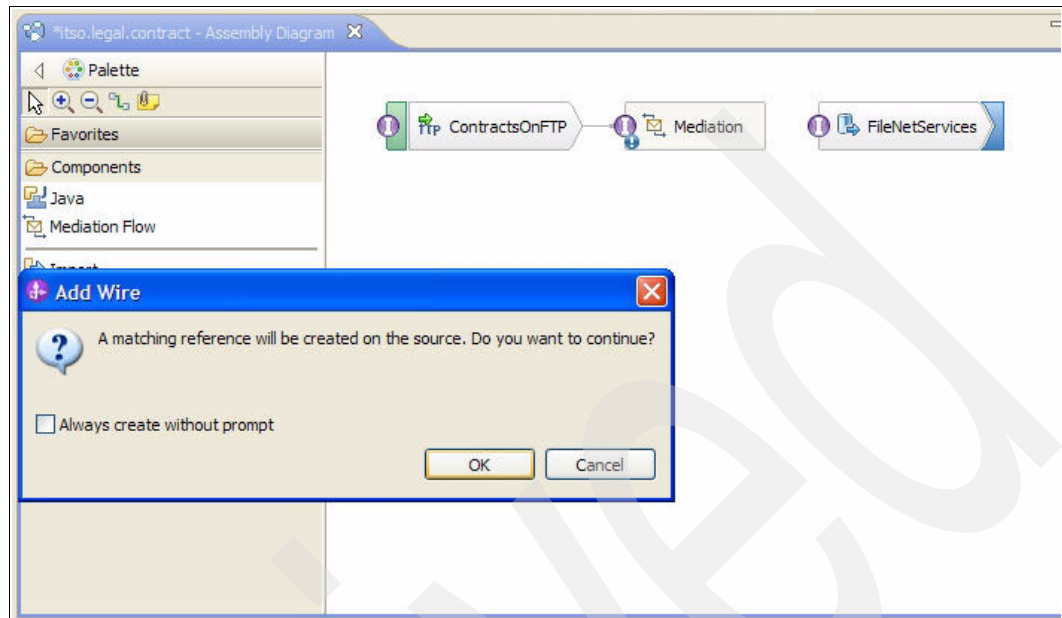


Figure 3-64 Wire the Mediation component and the FileNet service

20. Double-click the **Mediation** component to implement it. Click **Yes** for the confirmation window. Figure 3-65.

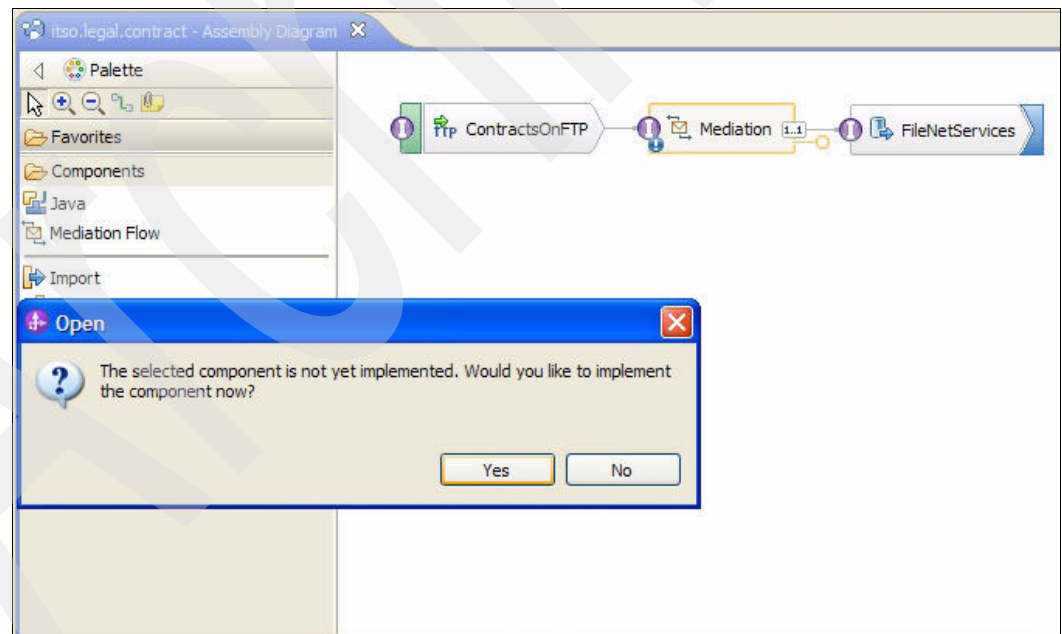


Figure 3-65 Start implementing the Mediation component

21. Choose a folder for the mediation implementation. You can optionally create a new folder, as well. For simplicity, we have selected the default root-level folder, as highlighted in Figure 3-66 on page 127.

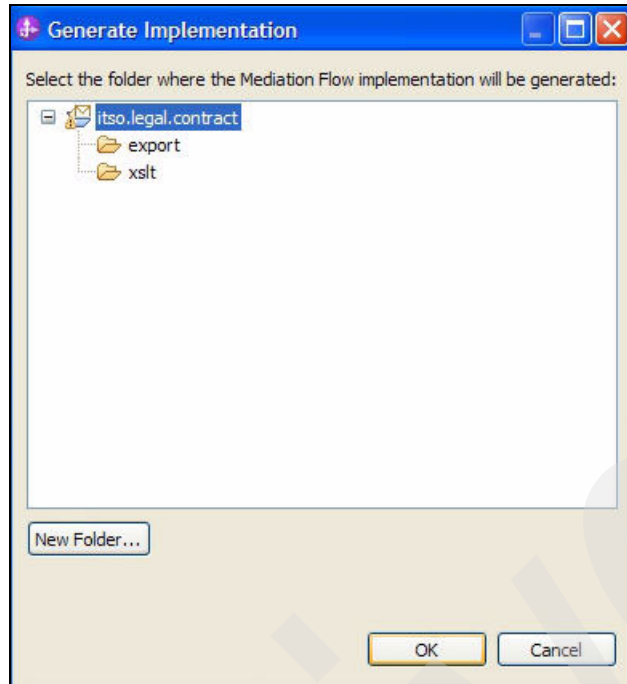


Figure 3-66 Choose a folder for the Mediation component's implementation

22. A new page appears for you to implement mediation. Select **emit** on the FTPEXport interface, as shown in Figure 3-67. This action opens a new page where you select **Service Integration**, as highlighted in Figure 3-67.

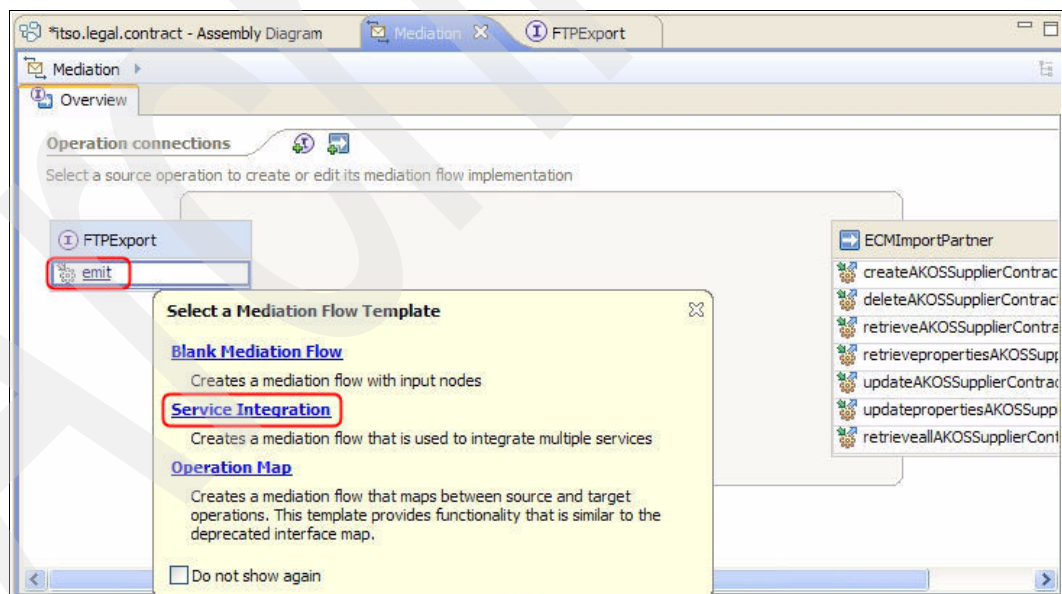


Figure 3-67 Integrate the FTP and FileNet services

23. When you select the Service Integration option, you must select **createAKOSSupplierContract** for the target operation, as shown in Figure 3-68 on page 128. Click **OK**.

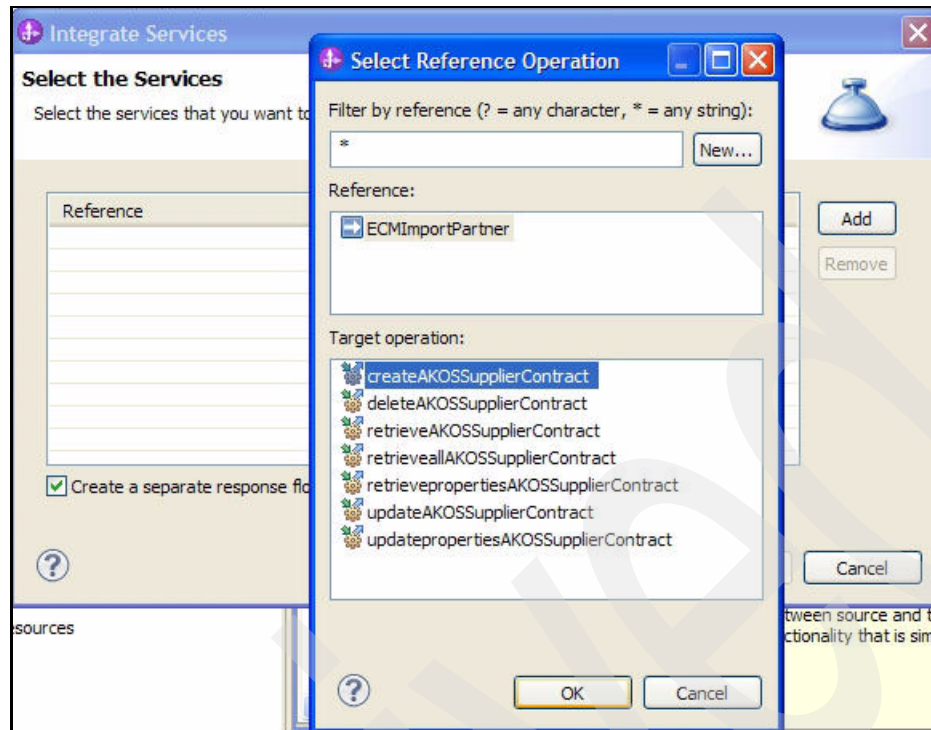


Figure 3-68 Select createAKOSSupplierContract for the target operation

24. Select the service that you want to integrate, as shown in Figure 3-69.

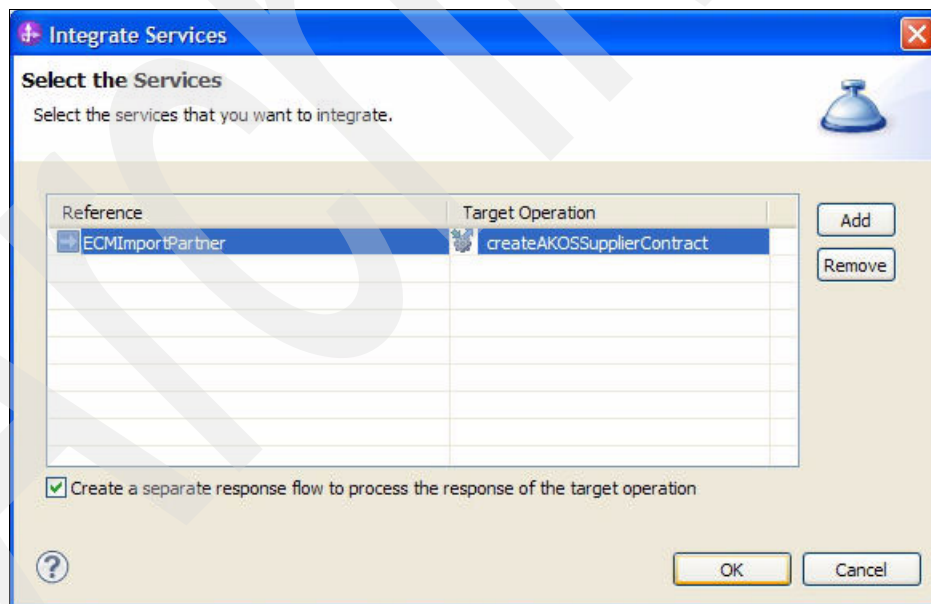


Figure 3-69 Select the service for integration

25. On the Request tab of the mediation module editor, wire the **FTPExport** service to the target **createAKOSSupplierContract** service. This action creates an XSLTransformation1 component, as shown in Figure 3-70 on page 129.

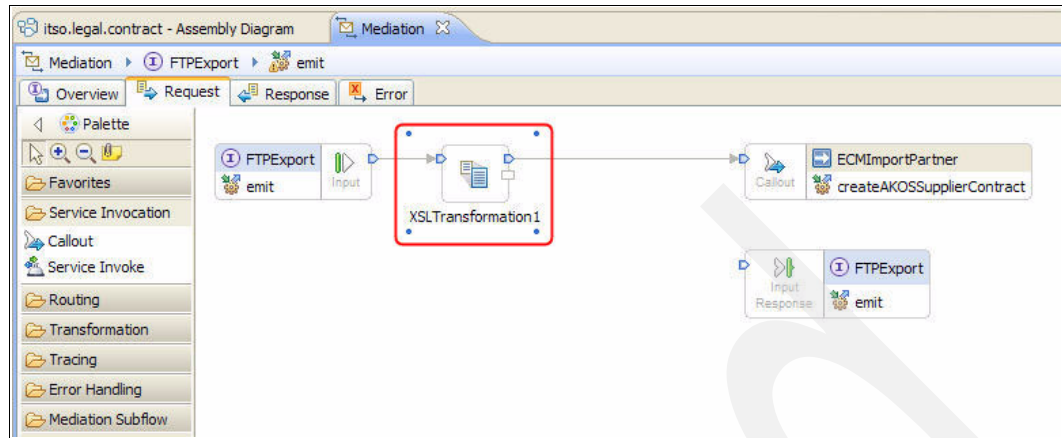


Figure 3-70 XSLTransformation1 component for mapping two separate formats

26. Double-click the **XSLTransformation1** component to open the map editor. IBM WebSphere Enterprise Service Bus provides excellent capabilities for mapping virtually any format to any other format. You can perform this entire mapping exercise in a few minutes, which is the greatest value-add for the clients. IBM performs the difficult work of managing all the data formats and their transformations and lets our clients concentrate on their business logic.
27. Open the **Properties** editor for this XSLTransformation1 component and select the **Default Values** tab. Select **Use target schema default/fixed value if one exists**, as shown in Figure 3-71.

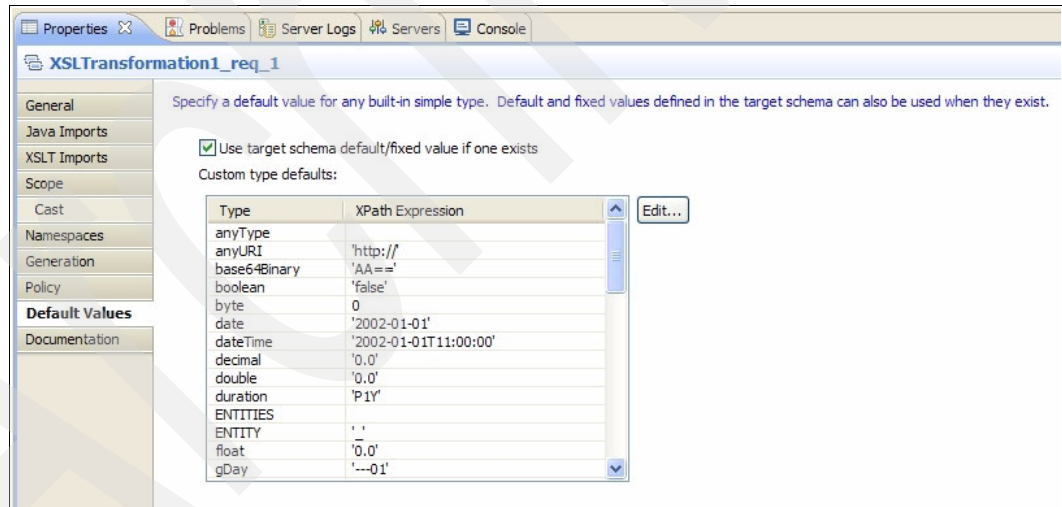


Figure 3-71 Specifying that the XSLTransformation1 component will use the target schema's default/fixed values

28. Simply drag and drop the source element to the appropriate target's element to create a map. Figure 3-72 on page 130 shows how the primarylegal element is mapped between the source and target services.

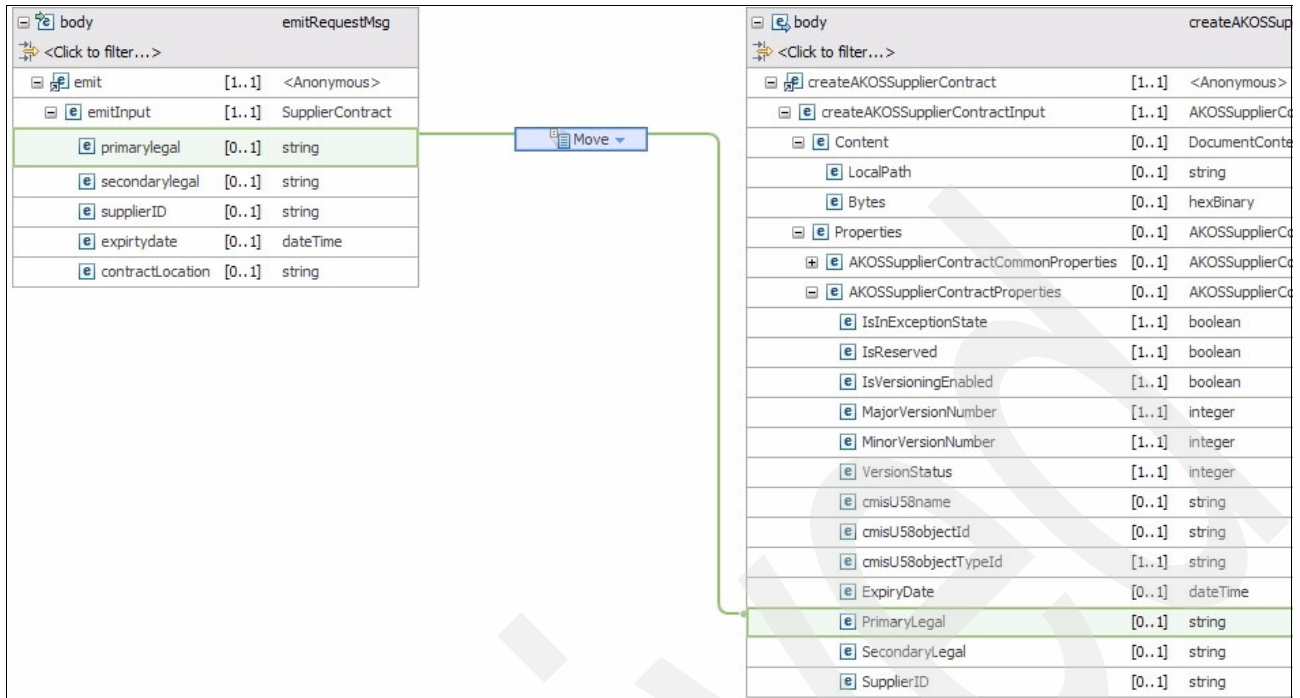


Figure 3-72 Map the elements of the source service and the target service

29. Repeat this mapping step to map all the source elements to the appropriate target elements.
30. However, in an actual client's environment, not all elements can be mapped one-to-one. Most of the time, adding extra logic around the map is required, for example, when assigning a default value for the element. In our case, RepositoryId on the target service does not have a mapping field in the source service, even though it is one of the mandatory properties for the FileNet service to work. To assign the default value for it, right-click the **RepositoryId** field and choose the **Create using default value** option. See Figure 3-73 on page 131.

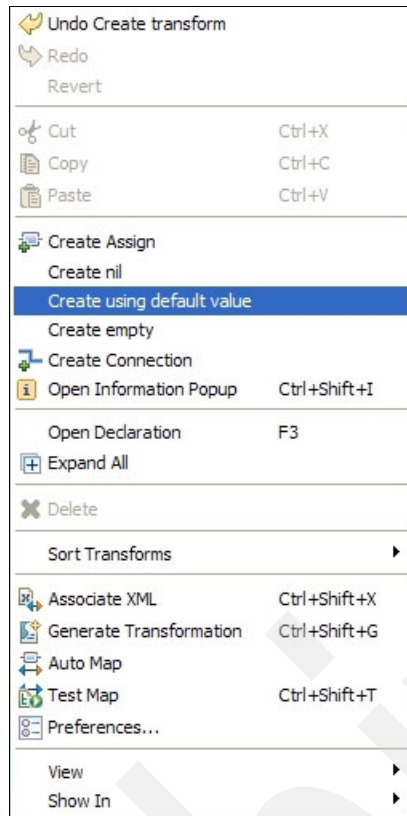


Figure 3-73 Assign a default value for the RepositoryId field

31. Repeat the previous step to assign a default values for every field that does not have a mapping in the source service but is required for the FileNet service to work.
32. Apart from this default value assignment, at times, it is required to assign one of valid values from the list, if the field is the enumeration type. VersioningState in the target service is of the enum type. Therefore, right-click **VersioningState** and choose the **Create Assign** option (Figure 3-74).

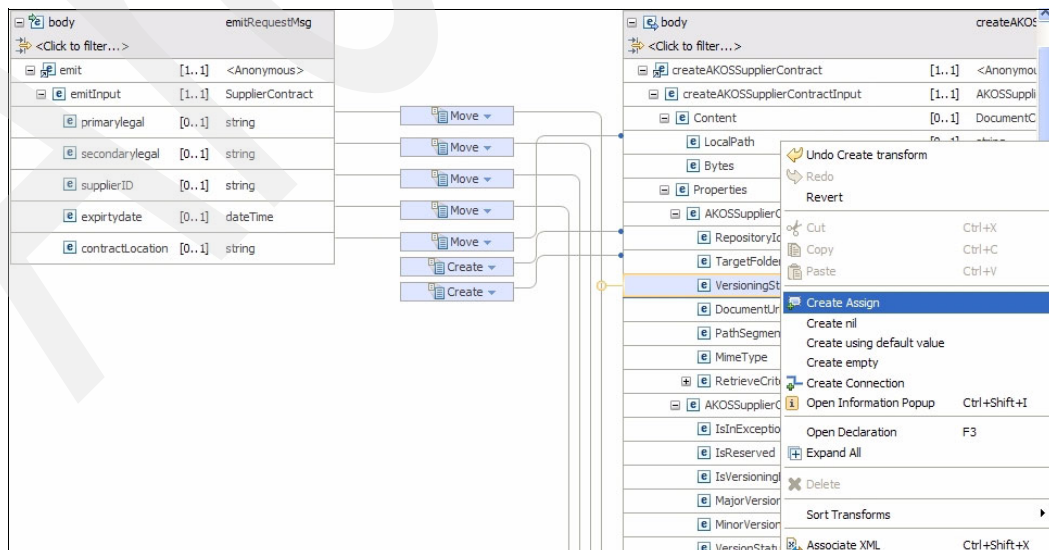


Figure 3-74 Assign the value for an enum field

33. After you have chosen the Create Assign option, an Assign node is created for that field. Select that **Assign** node and open the **Properties** view editor. On the first tab, which is the General tab, of the Properties view editor, for Value, type `major`, as shown in Figure 3-75.

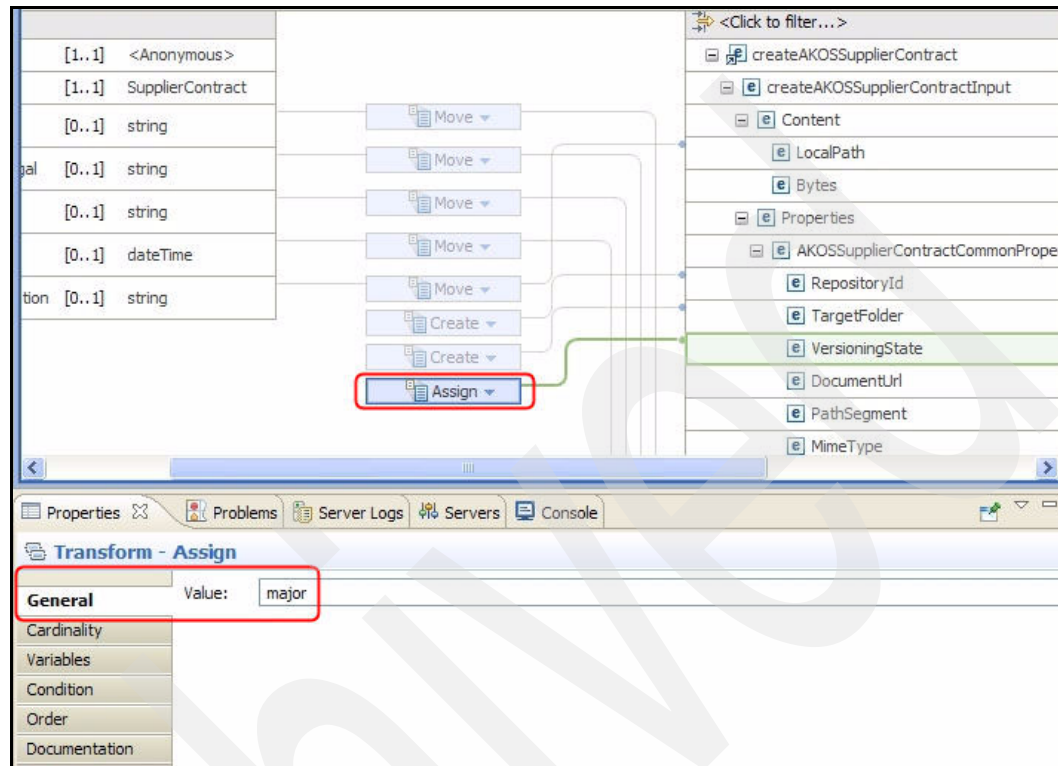


Figure 3-75 Assign a valid value for enum-type fields

34. The `cmisU58name` field on the target service is a required field, because it captures the title of the document to be generated on the FileNet system. Because the document titles must be unique, use the Concat function to concatenate the `primarylegal` and `supplierID` fields of the source to construct a unique value for the `cmisU58name` field by dragging and dropping the `primarylegal` field of the source service on the `cmisU58name` field of the target service. Again, drag the `supplierID` field of the source service and wire it to the same node. This action creates a Concat function, which concatenates the `primarylegal` and `supplierID` values to form the `cmisU58name`, as depicted in Figure 3-76 on page 133.

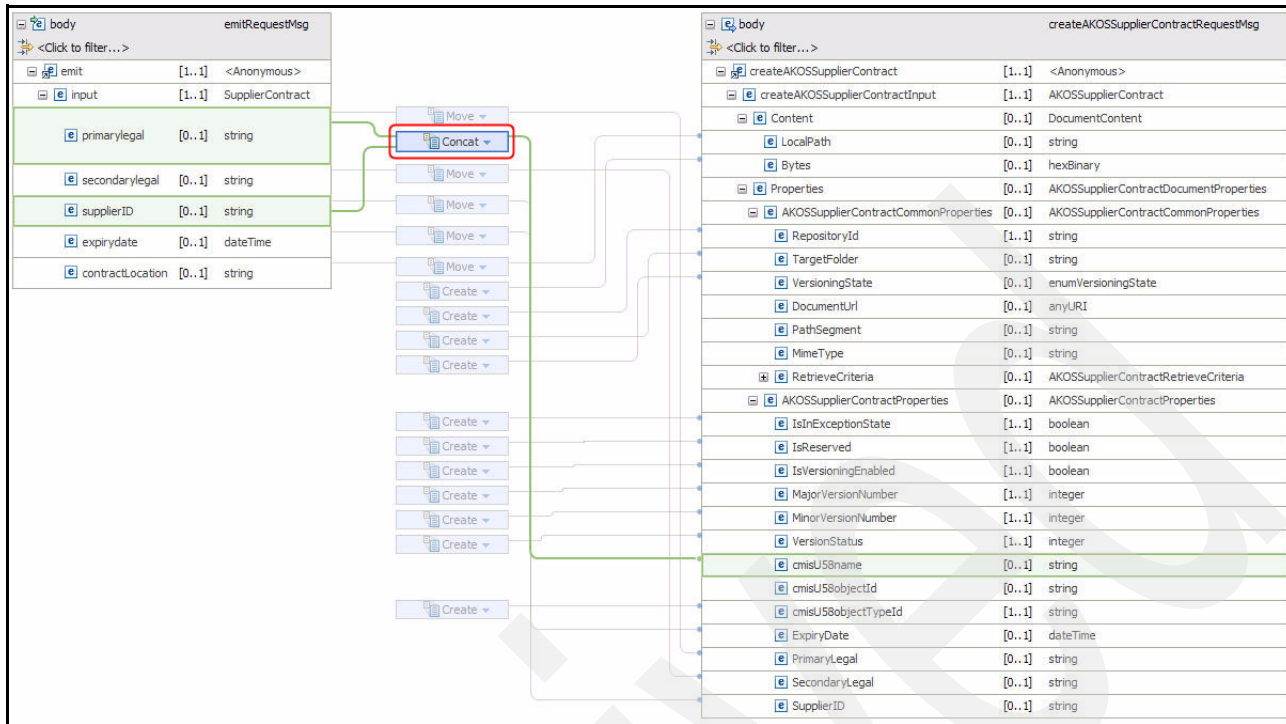


Figure 3-76 Concat function for constructing the cmisU58name field

- Wire back the response from the FileNet service to the FTP service. Open the **Response** tab of the Mediation component. Connect the Callout Response of **createAKOSSupplierContract** to the Input Response of the **emit** method, as shown in Figure 3-77.

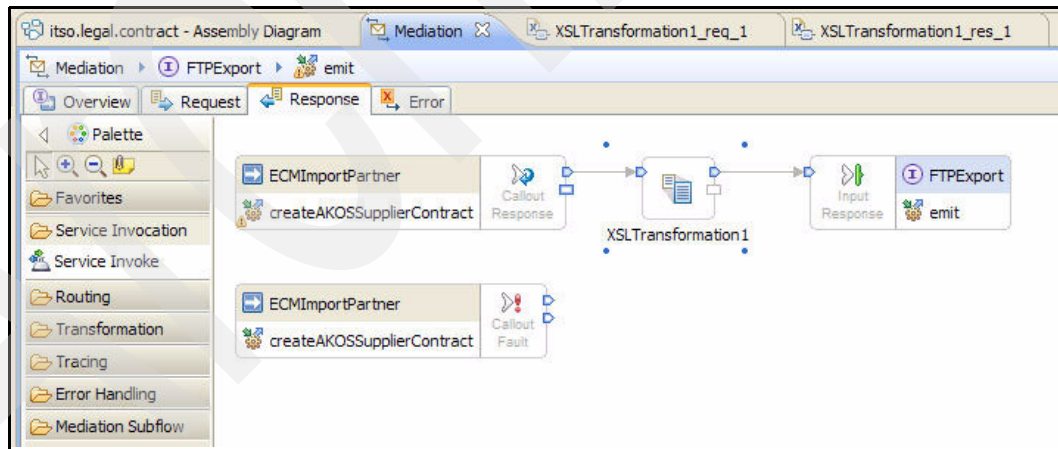


Figure 3-77 Wire the response back to the FTP service

- Double-click the **XSLTransformation1** component and map all fields one-to-one, because both the source and target formats are the same type.
- You have completed the integration between FTP and FileNet services. To validate the entire flow quickly, right-click the FTP service **ContractsOnFTP** and choose **Test Component**. Type the sample values for all input fields, and click the green invoke icon. Figure 3-78 on page 134 captures the sample values for all input fields.

Name	Type	Value
emitinput	SupplierContract	
primarylegal	string	rajan
secondarylegal	string	kumar
supplierID	string	mtds3987
expirydate	dateTime	2011-07-10T19:17:40.281+0530
contractLocation	string	c:/samplecontract.pdf

Figure 3-78 Sample values to validate the flow

38. After the entire flow executes successfully, the Integration Test Client displays various components through which the data has passed. Click the last **Return** node on the tree to see the final result of the execution. See Figure 3-79.

Integration Test Client: its0.legal.contract_Test

Events

This area displays the events in a test trace. Select an event to display its properties in the General Properties and Detailed Properties sections. [More...](#)

General Properties

Detailed Properties

Module: its0.legal.contract
Component: ContractsOnFTP
Interface: FTPExport
Operation: emit

Return parameters:

Value Editor | XML Source

Name	Type	Value
output	AKOSSupplierContract	
Content	DocumentContent	
Properties	AKOSSupplierContractDocum...	
AKOSSupplierContractCommonProperties	AKOSSupplierContractCommo...	
RepositoryId *	string	AKOS
TargetFolder	string	/ITSO/Legal/ContractDocuments
VersioningState	enumVersioningState <string>	major
DocumentURI	anyURI	http://9.184.166.126:9080/fncmis/resources/
PathSegment	string	rajanmtds3987(1)
MimeType	string	
RetrieveCriteria	AKOSSupplierContractRetriev...	
SelectClause	string	
WhereClause	string	
AKOSSupplierContractProperties	AKOSSupplierContractPropert...	
IsInExceptionState *	boolean	false
IsReserved *	boolean	false
IsVersioningEnabled *	boolean	true
MajorVersionNumber *	integer	1
MinorVersionNumber *	integer	0
VersionStatus *	integer	1
cmisUSName	string	rajanmtds3987
cmisUSObjectID	string	idd_E3E09F1C-970E-4188-96D5-CBE44BFBC4
cmisUSObjectTypeID *	string	
ExpiryDate	dateTime	2011-07-10T19:17:40.281+0000
PrimaryLegal	string	rajan
SecondaryLegal	string	kumar
SupplierID	string	mtds3987

Figure 3-79 Result of the final execution

39. To test the complete flow via the FTP channel, create a file containing the required supplier contract information and upload it to the target directory on the FTP server. Example 3-1 shows sample file content.

Example 3-1 Sample supplier contract file content

```
primarylegal=rajan
secondarylegal=kumar
supplierID=SRNG44
expirydate=2012-01-01T00:00:00.001Z
contractLocation=c:/samplecontract.pdf
```

Note: To keep this scenario simple, the scanned version of the supplier contract document is stored on the same machine where IBM WebSphere Enterprise Service Bus is running. You can extend this application by supplying the content via HTTP or through bytes.

40. After the file is successfully uploaded to the target FTP server, IBM WebSphere Adapter for FTP polls the file and processes it. Then, with the help of the Mediation component, the file is routed to the createDocument service of FileNet for documentation creation.
41. We have completed the process of implementing an FTP-based interface for the createDocument service.

3.4.4 Building a web application on the retrieveAllDocuments service to query FileNet documents

ITSO Enterprise has successfully built an application to store its supplier contract documents in FileNet. ITSO Enterprise wants to develop an application to view its supplier contract documents. ITSO Enterprise also requires a query-type interface where its employees can enter search criteria to fetch matching documents from FileNet.

Follow these steps to implement a web-based application to search the supplier contract documents by using the Web services binding technology of WebSphere Enterprise Service Bus:

1. Right-click the **Interfaces** section of the project and choose **New** → **Interface**, as shown in Figure 3-80.

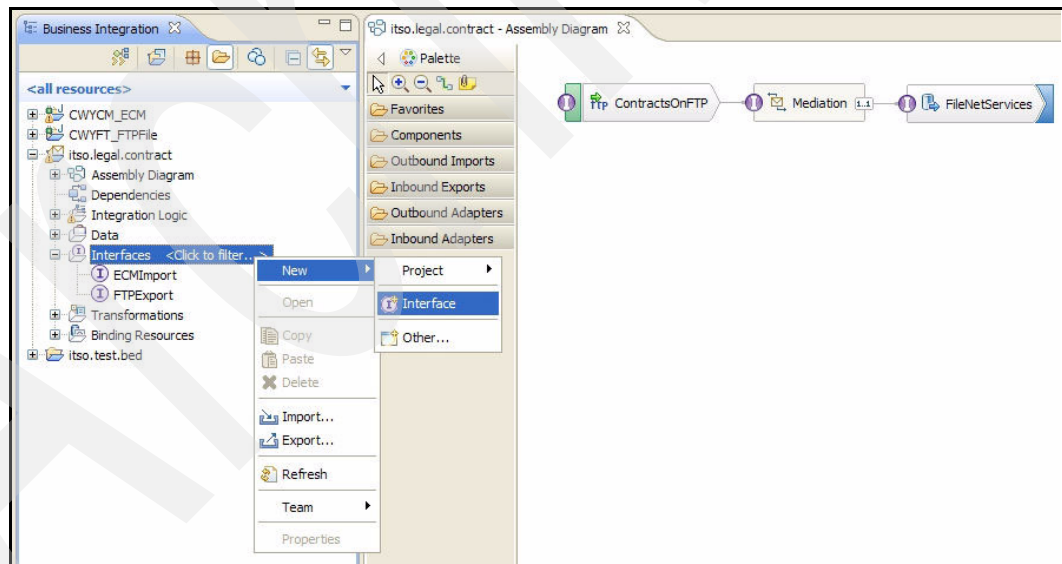


Figure 3-80 Create a new Interface

2. On the Create an Interface window, for the Name of the interface, type WSEExport and click **Finish**. See Figure 3-81 on page 136.

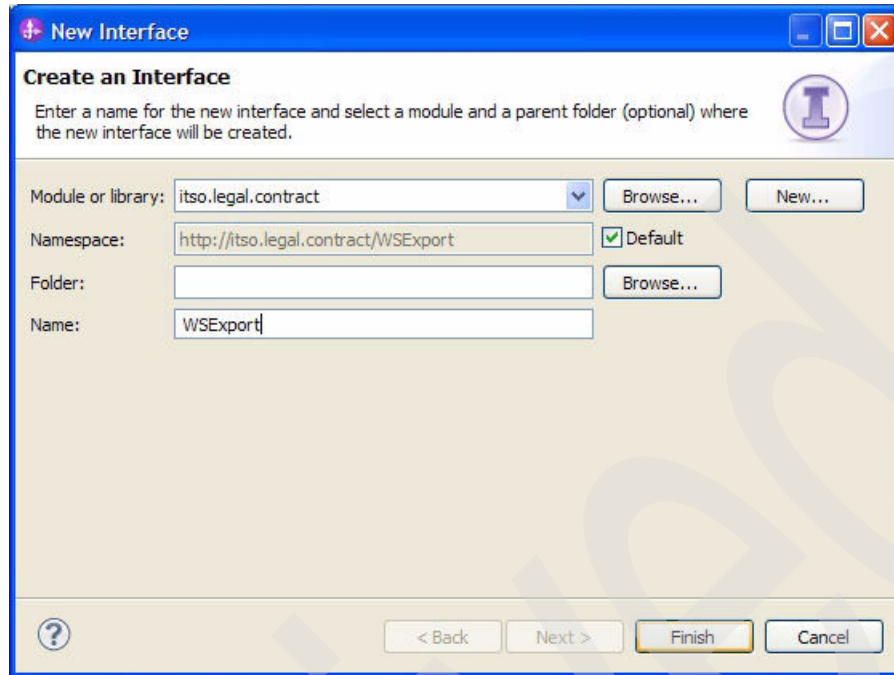


Figure 3-81 Enter the Name for the interface

- After the interface is created successfully, the interface editor automatically opens. Add a two-way operation called `queryDocument`. To keep it simple, name the input node `whereclause`, which is the string type, and name the output node `response`, which is the `AKOSSupplierContractContainer` type. The intention is to provide `whereclause` as a query to search the supplier contracts on the FileNet end. Matching documents will be returned as part of the container object. The container will have as many child business objects as the matching documents that are returned by FileNet. See Figure 3-82.

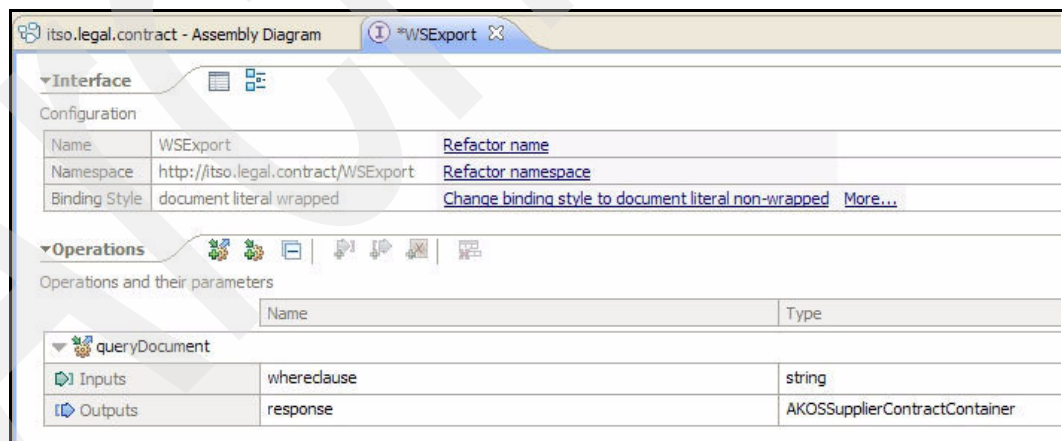


Figure 3-82 Interface with one operation `queryDocument`

- Next, provide the Web service binding capability to this interface and wire it to the existing mediation module to call the appropriate FileNet service via IBM WebSphere Adapter for Enterprise Content Management. To do so, drag and drop the **WSEExport** from the Interface section onto the Assembly Diagram. See Figure 3-83 on page 137. This action opens a new page where you choose **Export with Web Service Binding** and click **OK**.

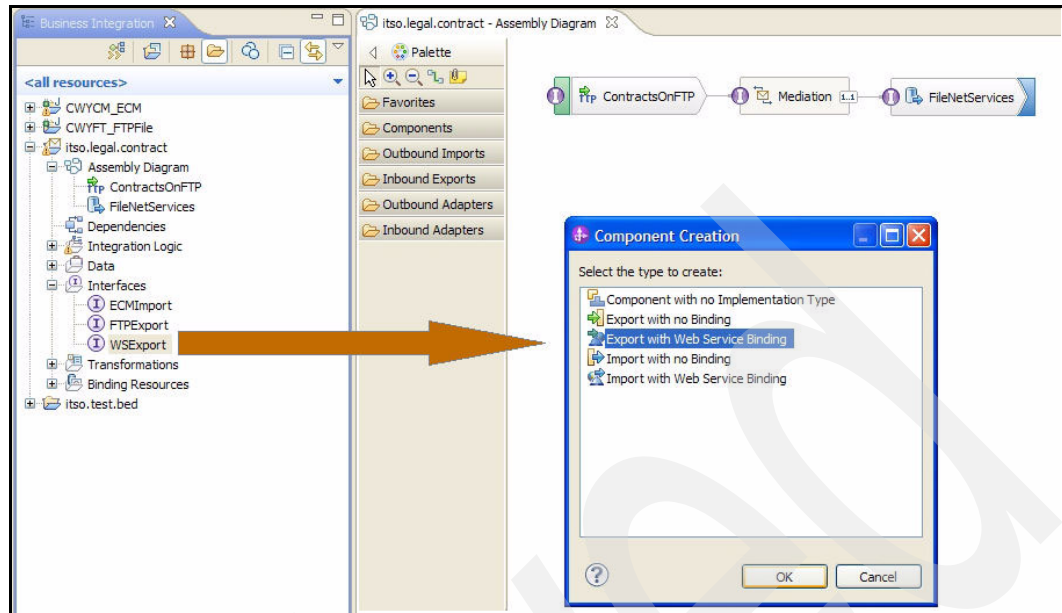


Figure 3-83 Drag and drop the WSEExport interface on the Assembly Diagram to start building the Web service binding

5. On the Select a Transport Protocol window, for the transport protocol, select **SOAP1.1/HTTP** for the Web service binding, and click **Next**. See Figure 3-84.

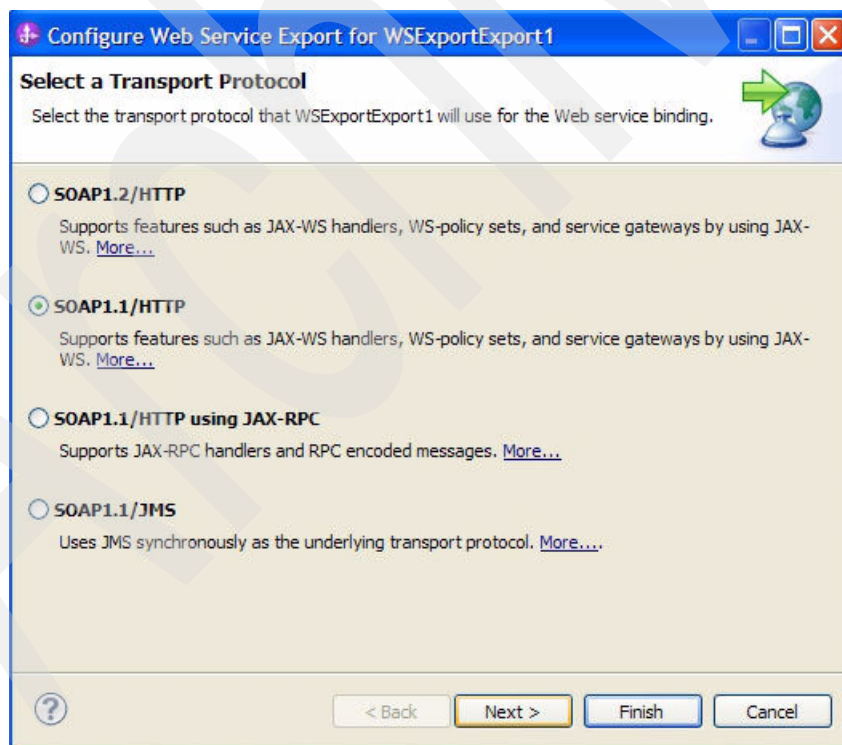


Figure 3-84 Choose the transport protocol for the Web service binding

6. Specify the target namespace for the Web service binding and click **Finish**, as shown in Figure 3-85 on page 138.

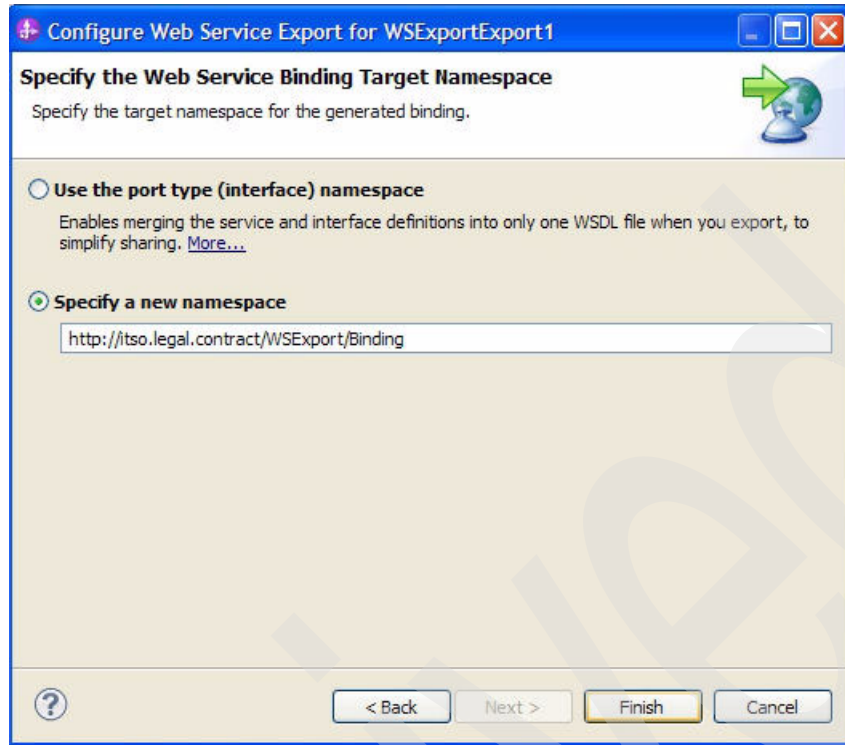


Figure 3-85 Provide the target namespace for the Web service binding

- The previous action adds the Web service export component on the Assembly Diagram. Wire **WSExportExport1** to the **Mediation** component that has been built already, as explained in Figure 3-86.

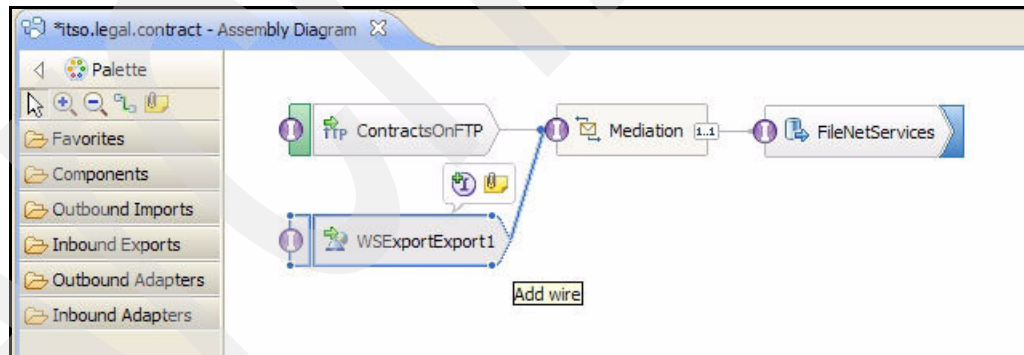


Figure 3-86 Wire the Web service component to the existing Mediation component

The Assembly Diagram looks like Figure 3-87 on page 139.

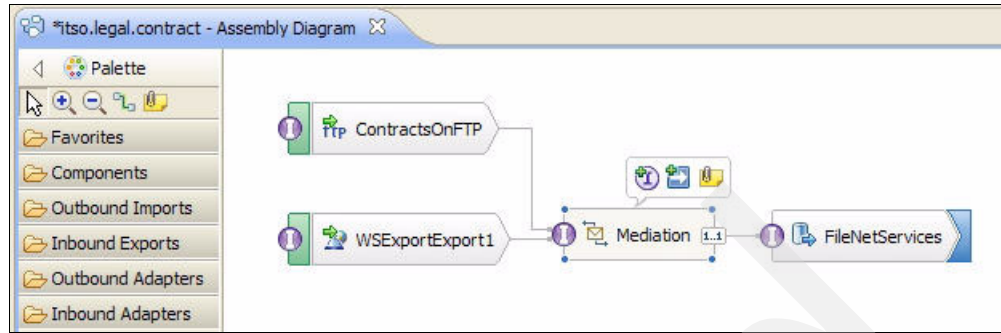


Figure 3-87 Assembly Diagram of all services

8. Double-click the **Mediation** module to extend its implementation for the Web Service binding component. Click the **add interface** icon, as highlighted in Figure 3-88, to add the newly created WSEExport interface to the mediation module.

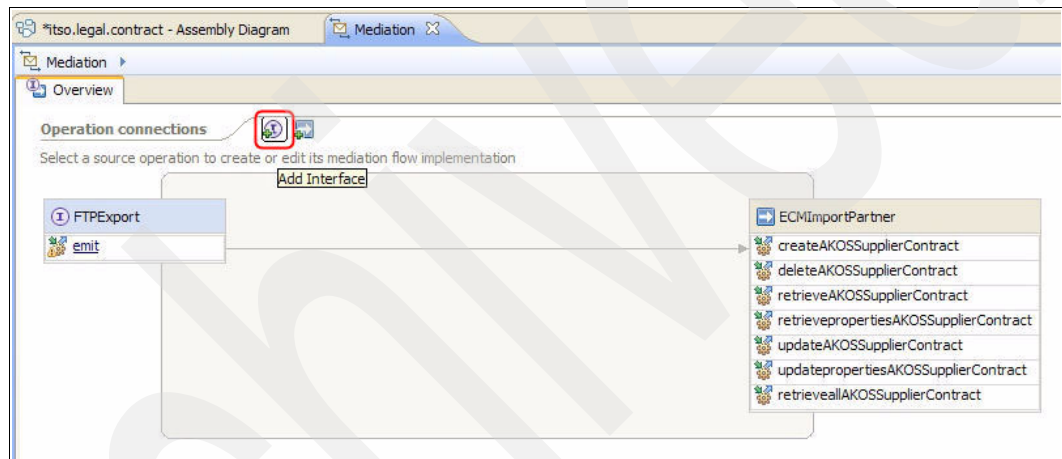


Figure 3-88 Add the newly created interface WSEExport to the mediation module

9. After you add the WSEExport interface to the Mediation component, click the queryDocument operation to select the Mediation flow template. Choose **Service Integration** as the mediation flow. See Figure 3-89 on page 140.

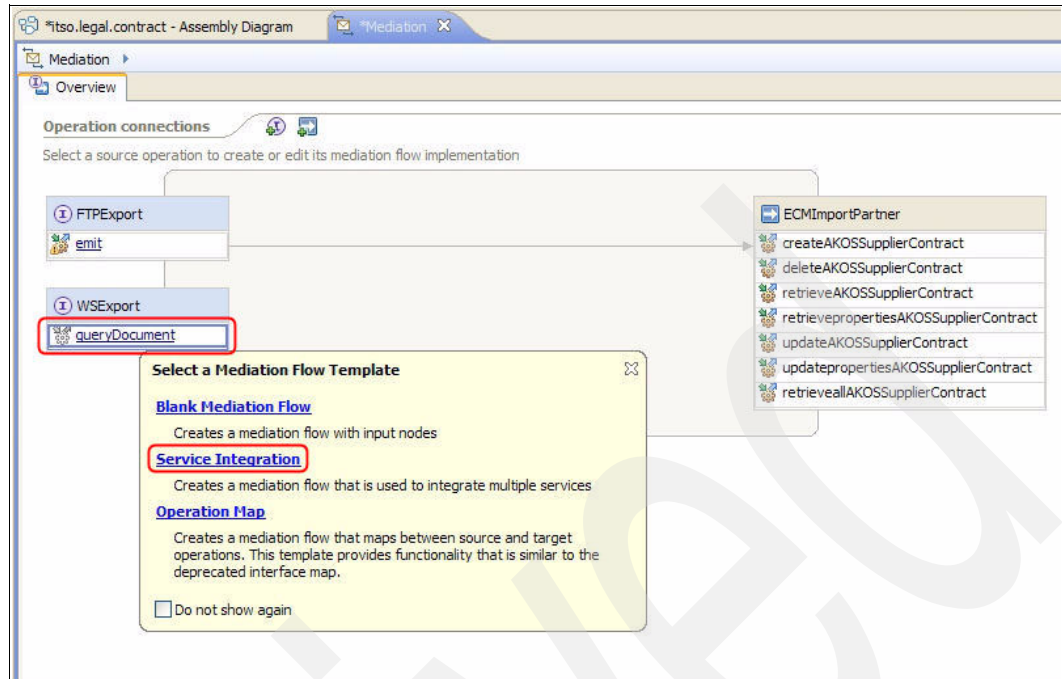


Figure 3-89 Choose Service Integration as the mediation flow

10. Add the reference to the target FileNet service. Add **ECMImportPartner** as a Reference, choose **retrieveallAKOSSupplierContract** as a Target Operation, and click **OK**. See Figure 3-90.

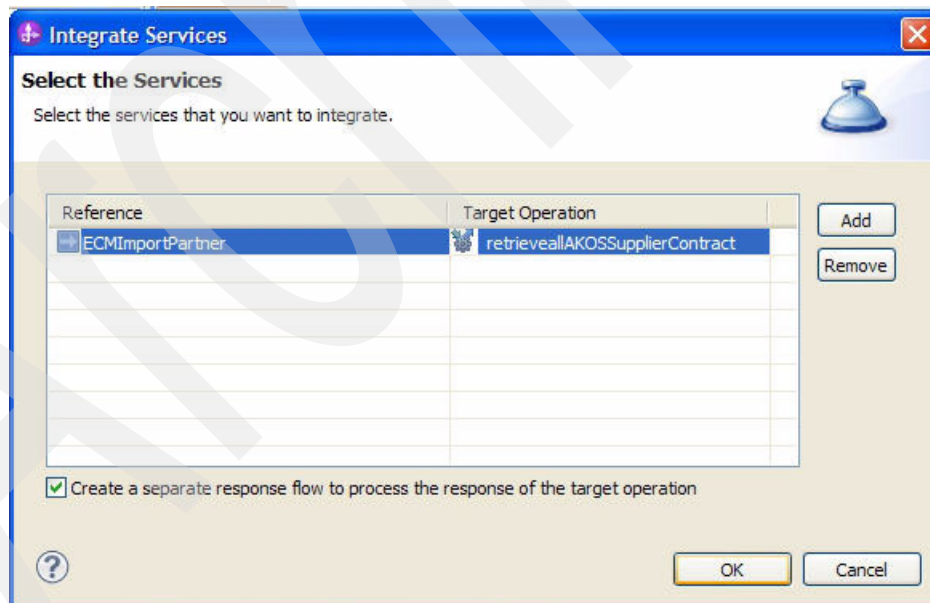


Figure 3-90 Choose a Reference and a Target Operation to integrate the services

11. The previous action creates the necessary mediation logic and places it on multiple tabs. On the Request tab, connect **queryDocument** and **retrieveallAKOSSupplierContract** using the XSLTransformation1 component, as shown in Figure 3-91 on page 141.

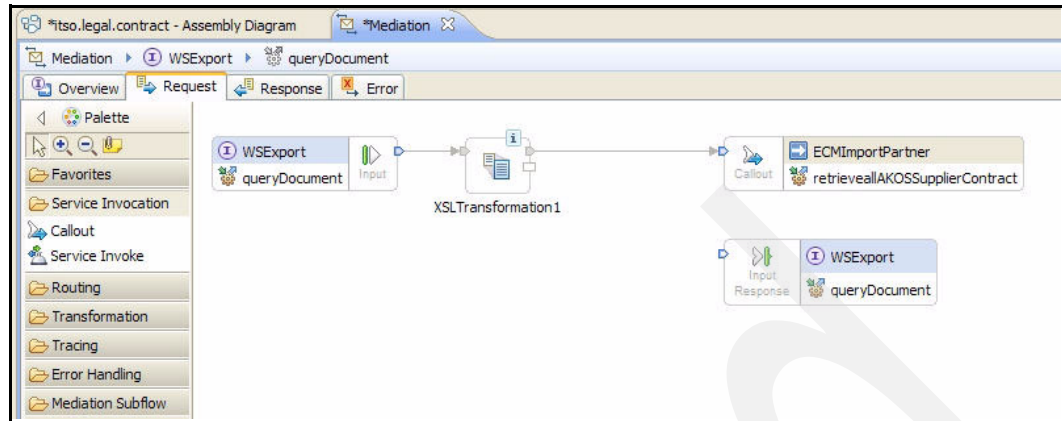


Figure 3-91 Wire the services using the XSLTransformation1 component

- Double-click the **XSLTransformation1** component to implement wiring the services. Enter a valid name for the map and click **Finish** on the next window. On the map editor, connect the **whereclause** of the queryDocumentRequestMsg to the **whereclause** of the retrieveallAKOSSupplierContract, as shown in Figure 3-92.

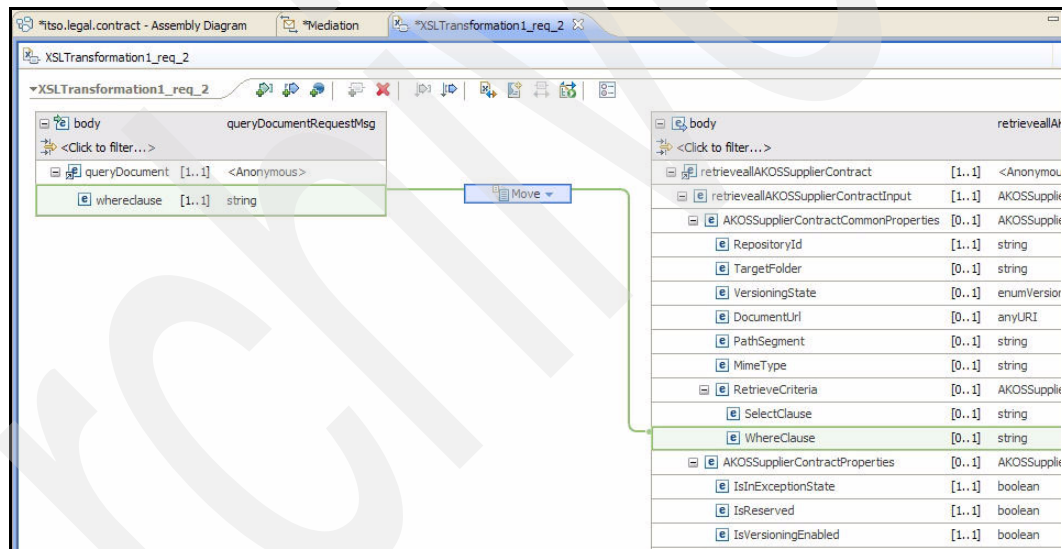


Figure 3-92 Map the necessary fields

- Follow similar steps to complete the response flow logic of the Mediation component. On the Response tab, connect retrieveallAKOSSupplierContract's response message to queryDocument's response. See Figure 3-93 on page 142.

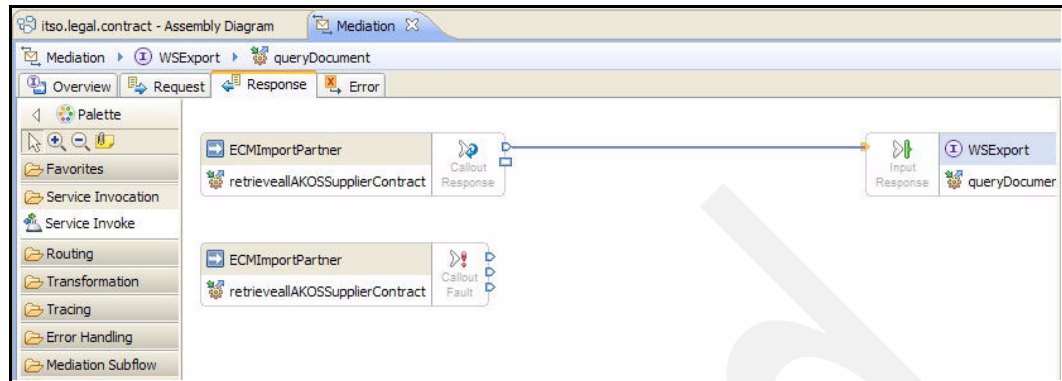


Figure 3-93 Connect the response flow of the Mediation component

14. Choose **XSL Transformation** as the type of mapping component for the response flow. Double-click the **XSLTransformation1** component. After naming the component, choose the **Auto map input to output** icon to automatically map all the input fields to the matching output fields. See Figure 3-94.

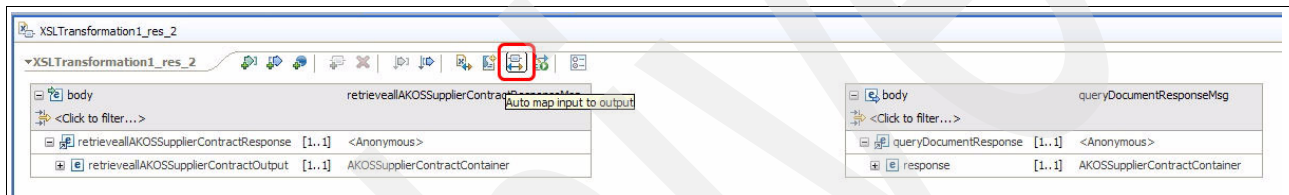


Figure 3-94 Auto mapping of input fields to output fields

15. On the next window, click **Finish**. Keep all the default values. See Figure 3-95 on page 143.

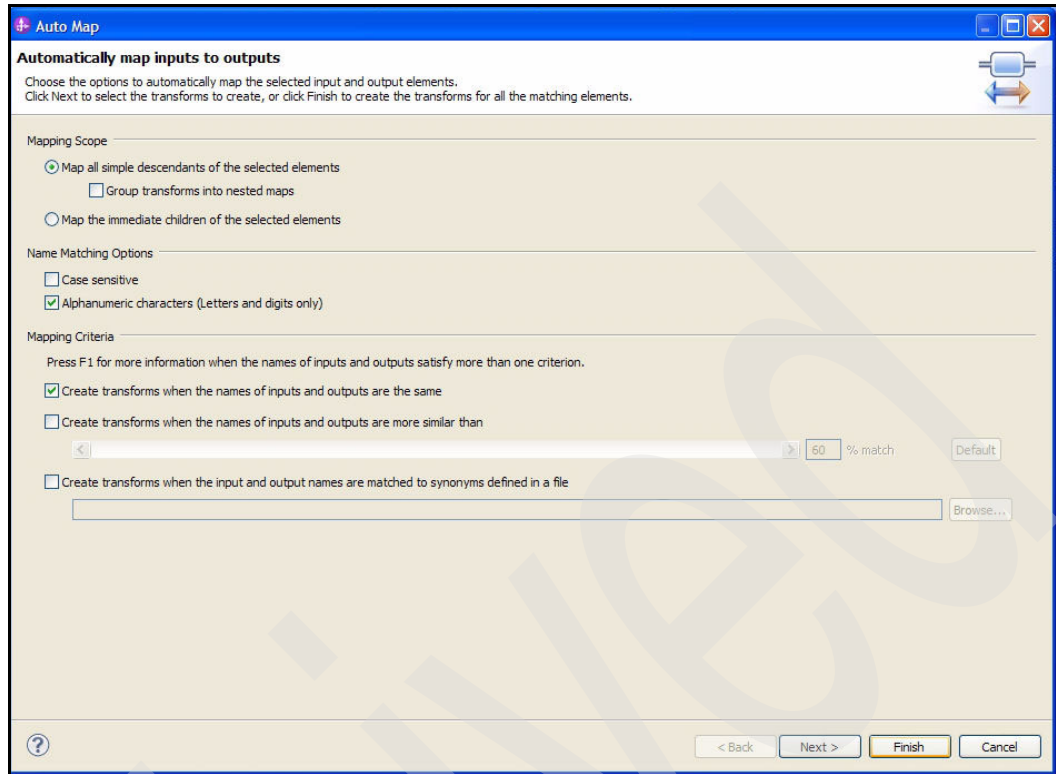


Figure 3-95 Configuration for advanced auto mapping functionality

The map looks like the map that is shown in Figure 3-96.

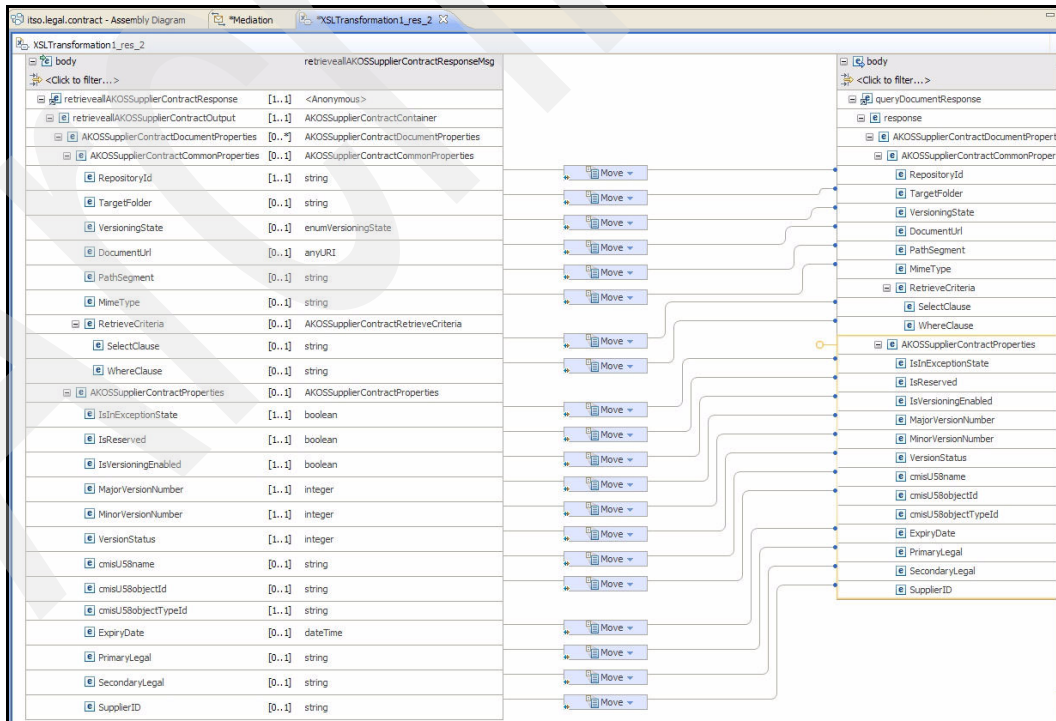


Figure 3-96 Final structure of the map with all the fields mapped

16. You have completed the Web service binding configuration. Next, you create a JavaServer Pages (JSP)-based Web service client and test it. To start the client generation process, right-click the corresponding Web Services Description Language (WSDL) component under the Web Service Ports section and choose **Web Services** → **Generate Client**. See Figure 3-97.

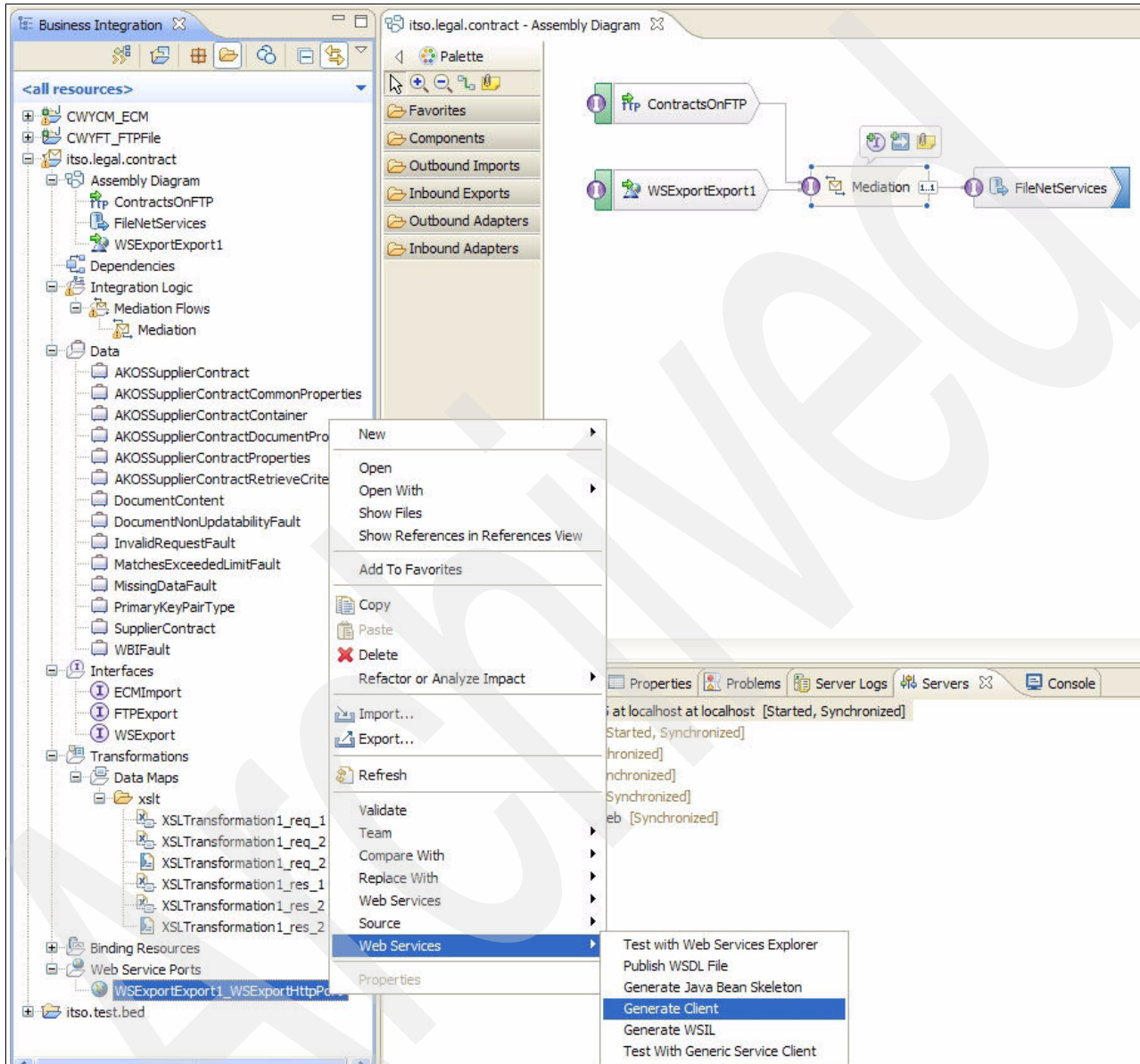


Figure 3-97 Launch the Web Service Client generation wizard

17. The Web Service Client generation wizard starts. On the first page of the wizard, for Client type, select **Java proxy** and move the client generation-level slider to Test client and click **Next**, as shown in Figure 3-98 on page 145.

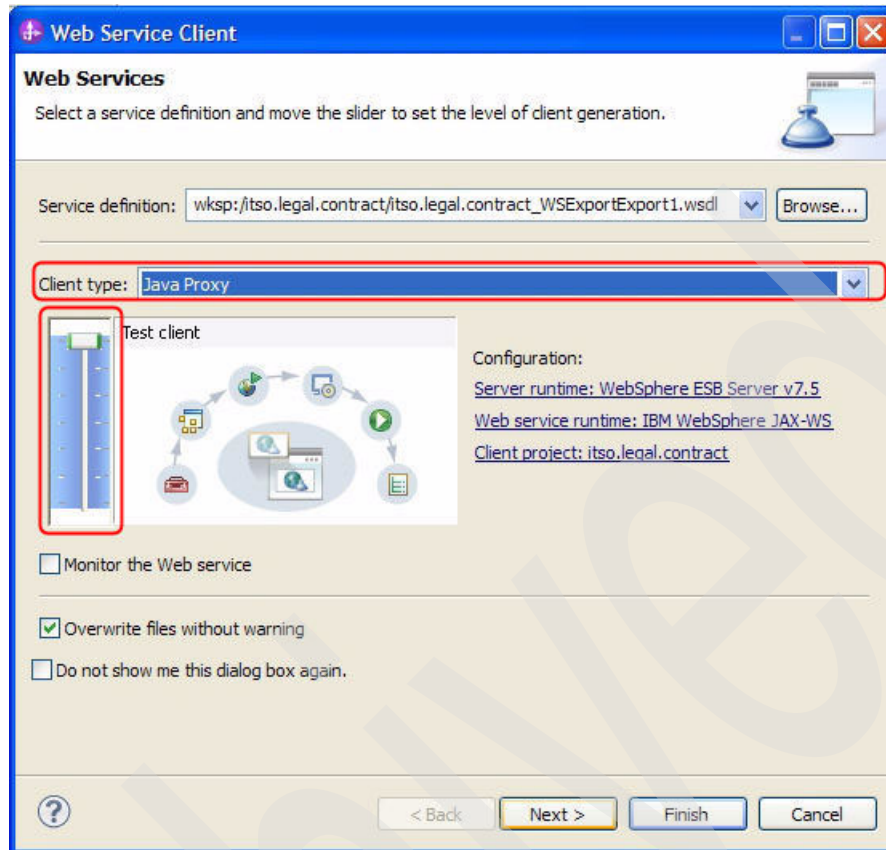


Figure 3-98 Web Service Client generation configuration

18. On the WebSphere JAX-WS Web Service Client Configuration wizard page, check **Generate portable client** and select **Generate ibm-webservicesclient-bnd.xmi template for overriding the service's endpoint URL**. Do *not* select the Enable MTOM option. Click **Next**. See Figure 3-99 on page 146.

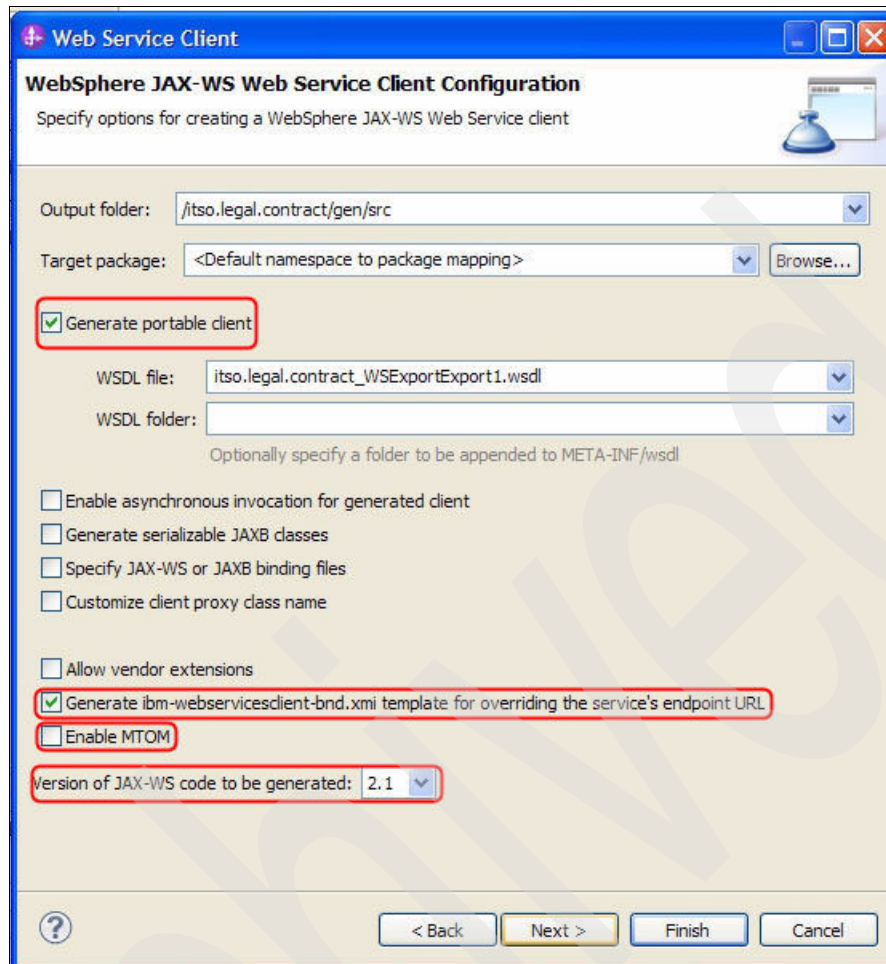


Figure 3-99 Additional settings for Web Service Client generation

19. On the Web Service Client Test page, for the Test facility, select **JAX-WS JSPs**.
20. Select the **_getDescriptor()** and **queryDocuments(java.lang.String)** methods for which to generate clients.
21. Click **Finish** to complete the Web Service Client generation. The wizard automatically generates the required files and updates them on the server (Figure 3-100 on page 147).

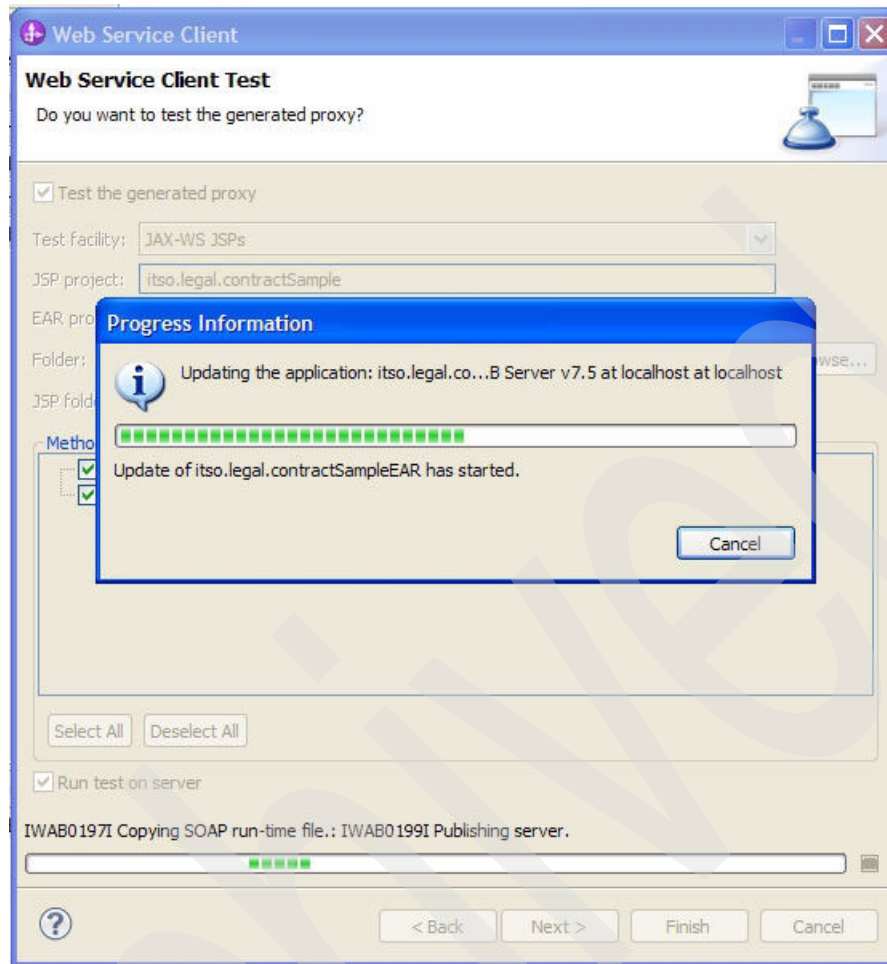


Figure 3-100 Generation of artifacts and code required for the client and updating them on the server

Resources: For more details about building the Web Service Client and testing it on IBM WebSphere Enterprise Service Bus, see these books:

- ▶ *Getting Started with IBM WebSphere Process Server and IBM WebSphere Enterprise Service Bus Part 1: Development, SG24-7608*
- ▶ *Getting Started with IBM WebSphere Process Server and IBM WebSphere Enterprise Service Bus Part 2: Scenario, SG24-7642*
- ▶ *Getting Started with IBM WebSphere Process Server and IBM WebSphere Enterprise Service Bus Part 3: Run time, SG24-7643*

3.4.5 Adding other protocols or application-based interfaces for other services

We have provided two interfaces for two of the FileNet services (FTP and HTTP). You can extend the capabilities further by adding other types of protocols or application-based interfaces for other services.

- ▶ Example 1: Integrate the Email application and the updateDocument service of FileNet.
- ▶ Example 2: Integrate the SAP Software retrieveDocument service of FileNet.

3.5 Summary

In this chapter, we explained the service enablement aspect of WebSphere Adapter for Enterprise Content Management and IBM FileNet. By using this solution, ITSO Enterprise was able to build applications to store and view its suppliers' legal contract documents on an on-demand basis. This solution was based on WebSphere Enterprise Service Bus using IBM WebSphere Adapter for Enterprise Content Management.

Service Reuse and Governance pattern

This chapter describes the Service Reuse and Governance pattern using the Smart Service-Oriented Architecture (SOA) approach. In this chapter, we discuss an ITSO Enterprise business scenario in which we reuse an enabled service across the internal infrastructure and demonstrate how to govern enabled services within the enterprise.

In this chapter, we use IBM WebSphere Enterprise Service Bus to reuse an enabled service and we introduce IBM WebSphere Service Registry and Repository (WSRR) for the governance of service.

This chapter has the following sections:

- ▶ 4.1, “Introduction to the business scenario” on page 151
- ▶ 4.2, “Architectural diagram” on page 153
- ▶ 4.3, “Benefits” on page 153
- ▶ 4.4, “Summary” on page 198

How to recreate these scenarios: You can download the configuration files and programs that are used in this chapter from the ITSO Redbooks FTP site. You can use these files to recreate these scenarios in your environment. For download instructions, refer to Appendix A, “Additional material” on page 401.

Archived

4.1 Introduction to the business scenario

Based on the success of accessing a single service across departments with the customer information service, as described in Chapter 2, “Service Enablement pattern” on page 23, ITSO Enterprise begins to evaluate other problems that an SOA approach can solve, such as integrating multiple services across the departments.

4.1.1 Scenario problem

After placing an order, if the customer wants to find out the status of an order, the customer makes a phone call to the Customer Sales Representative (CSR) team to get the latest status of the order. But the CSR team does not have direct access to the ITSO Enterprise ordering system. So, the CSR has to put the call on hold and phone the ITSO Enterprise Orders team to query the order for a given customer, as described in Figure 4-1. The customer gets frustrated after being put on hold for the phone call to query the order by the CSR.

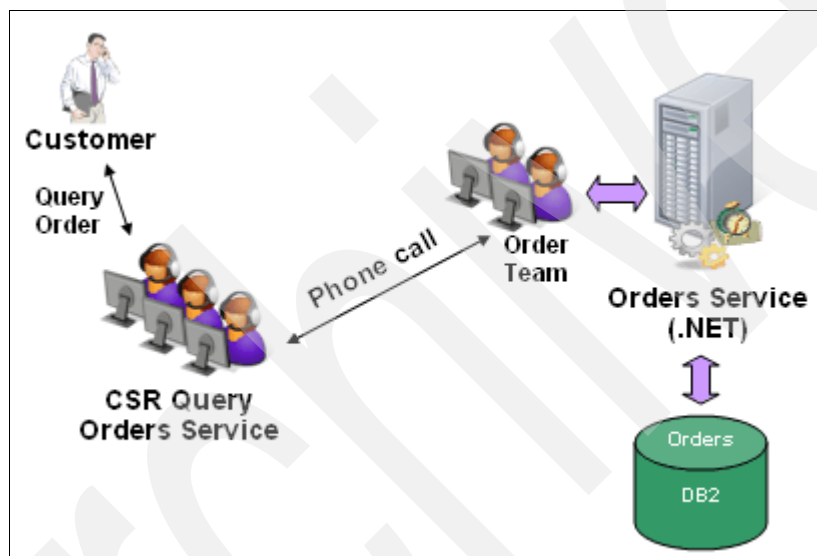


Figure 4-1 ITSO Enterprise current query order service

After getting many complaints from the customers, the CSR team goes to the Orders team and asks the Orders team to make the ITSO Enterprise ordering system available to the CSR team as a cross-department service.

The members of the Orders team are convinced of the need for the CSR team to access the order service, because the Orders team sees the cost of all the calls from the branch staff and also realizes that ITSO Enterprise must provide better customer service to increase customer satisfaction.

The order system was implemented in .NET and they are happy to maintain the work required to expose an interface to it. The Orders team chooses a Microsoft Windows Communication Foundation (WCF)-based Web service that is hosted in Internet Information Services (IIS), because this method is the most natural technology choice for the team.

However, the Orders team members are concerned that if they make the .Net system available as a service outside of their department, more teams will use it and the Orders team has not planned for enough capacity for that many users.

4.1.2 Scenario solution

IBM helps ITSO Enterprise to solve the problem by showing the company how to reuse the ordering system with the help of IBM WebSphere Enterprise Service Bus (ESB). Also, IBM helps ITSO Enterprise by describing how to build a service gateway using WebSphere Service Registry and Repository, as described in Figure 4-2.

WebSphere Service Registry and Repository enforces a service-level agreement (SLA) between the CSR and Orders teams. Also, it prevents other internal users from using the Orders system without first establishing their own SLAs.

This solution consists of a Gateway pattern in IBM WebSphere Enterprise Service Bus, which is linked to WebSphere Service Registry and Repository.

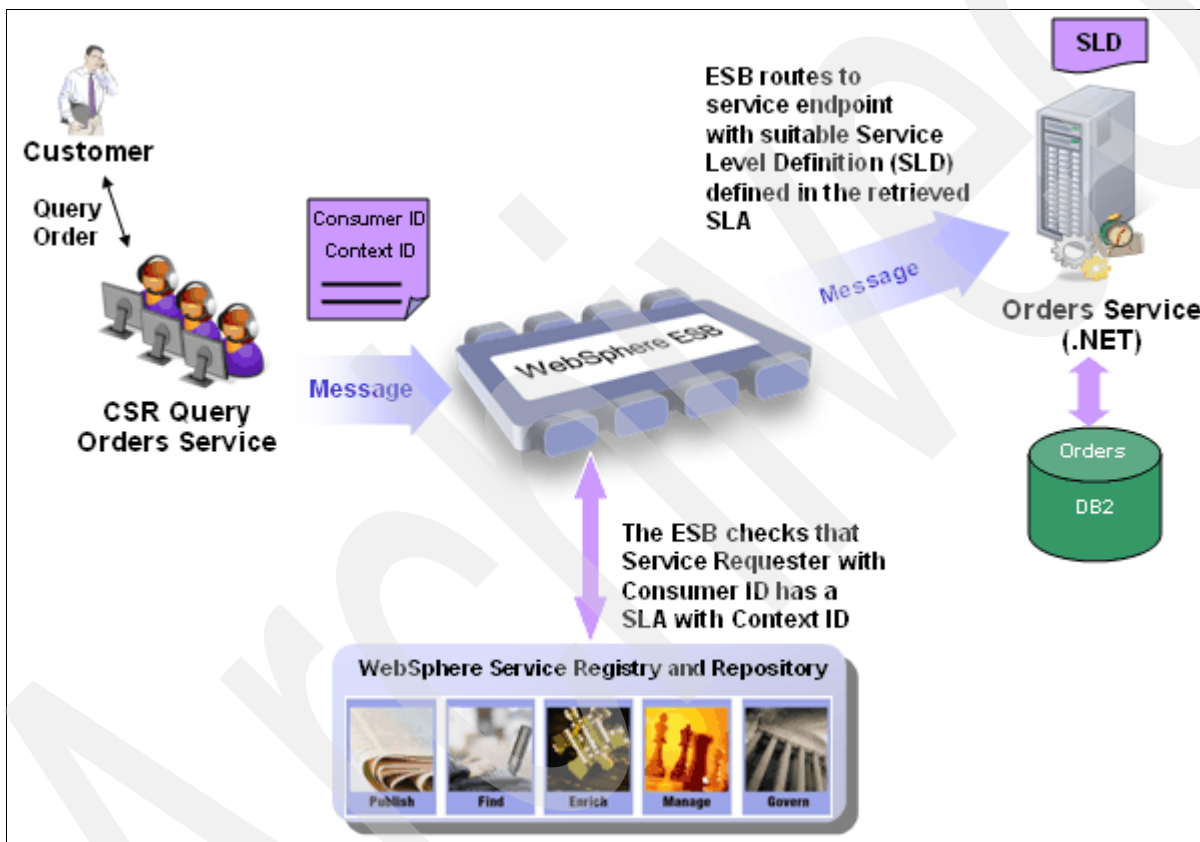


Figure 4-2 WebSphere ESB and WebSphere Service Registry and Repository Solution for ITSO Enterprise

Using the IBM solution, we see that without the SLA in place, the CSR application cannot access the service and it cannot access it directly in the .Net service, because a secure link is established. Only IBM WebSphere ESB is able to connect to the IIS server where the .Net service is hosted.

In IBM WebSphere Service Registry and Repository, we add the SLA for the CSR service. The CSR application is able to view the orders that are associated with each customer within the CSR application.

4.2 Architectural diagram

In this scenario, we use the following products:

- ▶ IBM Integration Designer V7.5
- ▶ IBM WebSphere ESB Server V7.5
- ▶ IBM WebSphere Service Registry and Repository V7.5
- ▶ IBM DB2 database V9.7

Figure 4-3 shows an architectural diagram of an ITSO Enterprise with IBM WebSphere ESB and WebSphere Service Registry and Repository solution.

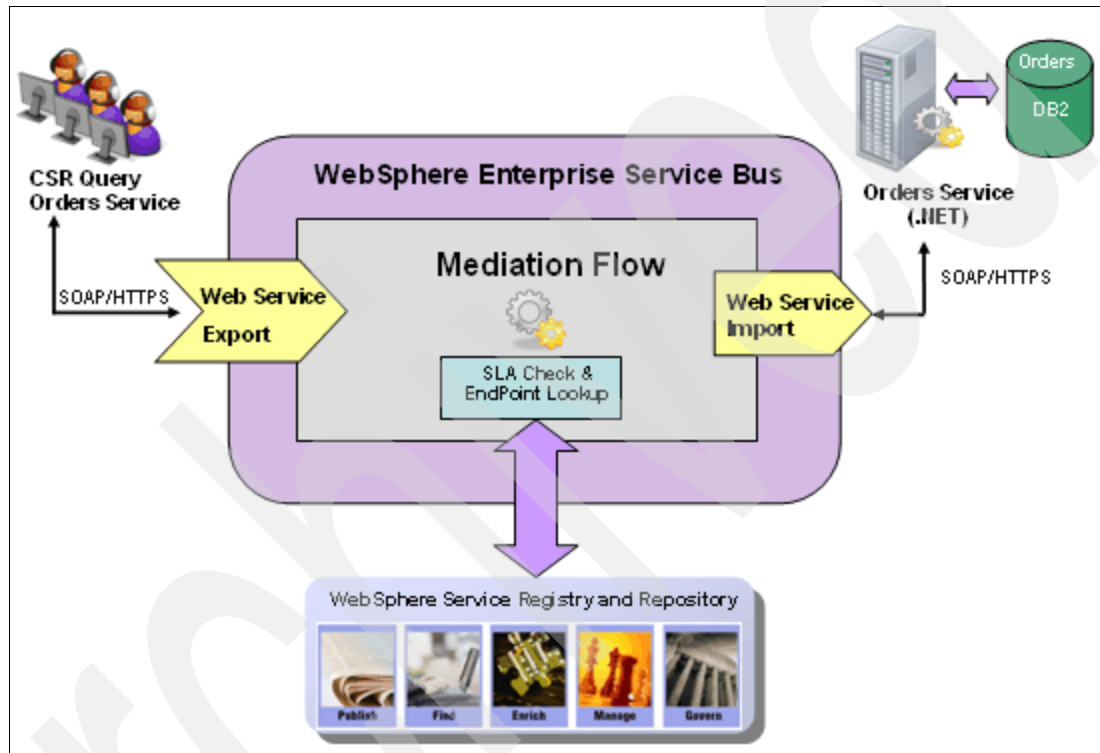


Figure 4-3 Assembly Diagram of WebSphere ESB and WebSphere Service Registry and Repository solution

Process: We expose a Web service from the mediation module as a Web service export to receive the SOAP request message from the CSR query orders service. We use the SLAEndpoint lookup to WebSphere Service Registry and Repository in the mediation flow and route the request by calling the Orders service (.Net) as a Web service import. The generated SOAP response message is routed to the CSR query orders service.

4.3 Benefits

An SLA can be a formal or informal contract that defines service priorities and guarantees. Many companies want to manage their services by exploiting SLAs. For example, if a service can only process so many messages per second, you can agree to a certain number of messages with specified consumers.

WebSphere Service Registry and Repository stores SLAs as part of its governance enablement profile. A consumer service representation in WebSphere Service Registry and Repository can have an SLA that describes the contract to a provider service definition. This SLA can be checked by WebSphere ESB to determine whether a consumer is entitled to invoke a service. Additional metadata that is stored in the SLA describes the agreed-to parameters in which the consuming service is allowed to operate, for example, the level of service, minimum and maximum messages per day, and so on.

For more information, refer to this website:

http://publib.boulder.ibm.com/infocenter/esbsoa/wesbv7r5/topic/com.ibm.websphere.wbpm.scenarios.esb1.doc/topics/cscn_sla.html

WebSphere Service Registry and Repository offers the following value propositions:

- ▶ Enable the SOA governance of services throughout the service life cycle
- ▶ Promote reuse and eliminate redundancies by increasing the visibility of services, applications, and processes
- ▶ Enhance connectivity by increasing the runtime flexibility of applications that are integrated using the ESB
- ▶ Optimize the use of services in SOA by exchanging rich service information with runtime monitoring tools and operational data stores
- ▶ Enable governance and life-cycle management of the organization's high-value applications, such as WebSphere MQ, IBM CICS®, and IMS
- ▶ Enable policy management across the SOA life cycle, spanning all domains of policy

WebSphere Service Registry and Repository offers the following potential benefits:

- ▶ Encourage reuse: WebSphere Service Registry and Repository improves the enterprise-wide visibility of a service that reduces redundancies, eliminates duplication, and encourages reuse.
- ▶ Enhance connectivity: WebSphere Service Registry and Repository increases the runtime flexibility of applications and processes by allowing ESBs to select services based on service metadata and enforce policies that are associated with the services and by managing multiple service versions.
- ▶ Enable Governance: WebSphere Service Registry and Repository provides a trusted and centralized source of services and related metadata and policies, by governing the life cycle of service and policies

For more information about WebSphere Service Registry and Repository V7.5, see this website:

http://www-01.ibm.com/software/integration/wsrr/features/index.html?S_CMP=wspace

4.3.1 Technical implementation

This section is divided into two sections that explain in step-by-step instructions how to implement the solution using the two products: WebSphere ESB and WebSphere Service Registry and Repository. The first part describes enabling governance in WebSphere Service Registry and Repository; defining the service consumer, the service provider, and all the artifacts that are associated with them. The second part describes the construction of a WebSphere ESB mediation flow that allows endpoint selection that is based on a service consumer to service provider relationship via an SLA.

4.3.2 Enabling governance in WebSphere Service Registry and Repository for contract enforcement

In this section, we register and govern provider and consumer services for the ITSO Enterprise company and define a contract between the consuming organization's service and the providing organization's service. This section contains the following topics:

- ▶ "Activating the Governance Enablement Profile"
- ▶ "Registering the Web Services Description Language" on page 156
- ▶ "Identifying the business capability through a business service" on page 157
- ▶ "Realizing the business service with a service version" on page 161
- ▶ "Creating an SLD for the service version" on page 166
- ▶ "Activating the service endpoint and completing the service life cycle" on page 170
- ▶ "Registering and governing the service consumer" on page 172
- ▶ "Creating an SLA between the service consumer and the service provider" on page 172

Activating the Governance Enablement Profile

The Governance Enablement Profile is a configuration for a WebSphere Service Registry and Repository (WSRR) instance and contains a well-defined service data model, service taxonomy, roles, policies, and user interface configuration perspectives that are derived from preferred practices from working with clients. You must load and activate the Governance Enablement Profile manually after you have completed the installation of WebSphere Service Registry and Repository. You can have multiple configuration profiles for a WSRR instance, but only one configuration profile can be active.

A WebSphere Service Registry and Repository configuration profile contains a complete set of WebSphere Service Registry and Repository configuration files. Configuration profiles are used for the backup, restore, and management of entire sets of WebSphere Service Registry and Repository configuration.

A WebSphere Service Registry and Repository configuration profile contains these components:

- ▶ **Web UI views:** To display registry content in a way that is suitable for its intended use
- ▶ **Roles and perspectives:** User roles, and web UI perspectives to support those user roles
- ▶ **Classification systems:** To support your user-defined models, business domains, and technical domains of relevance
- ▶ **Life cycles:** Appropriate to the service life cycle and its governance
- ▶ **Configuration of user-defined validators, modifiers, and notifiers:** To implement governance policies

Configuration profiles are loaded into WebSphere Service Registry and Repository from a configuration profile compressed file. WebSphere Service Registry and Repository provides the following profiles:

- ▶ Basic profile (BasicProfile_v75.zip)
- ▶ Governance enablement profile (GovernanceEnablementProfile_v75.zip)

The profiles are located in the `<WAS_INSTALL_ROOT>\WSRR\config` directory.

Follow these steps to load and activate the Governance Enablement Profile:

1. Log on to the WebSphere Service Registry and Repository Web user interface by entering `http://hostname:port/ServiceRegistry/` into the address bar of the web browser.

WebSphere Service Registry and Repository Web user interface URL: The *hostname* is the IP address or host name of the WebSphere Application Server that hosts the WebSphere Service Registry and Repository application, and the *port* is the server port number (the default port number is 9080 and the secured port number is 9443). If security is enabled, the browser directs you to the secured URL address.

2. Switch to the Configuration perspective by selecting the **Configuration** perspective from the **Perspective** drop-down menu.

Configuration perspective: The WebSphere Service Registry and Repository web user interface supports the concept of perspectives. Various perspectives can provide separate views of the data that is stored in WebSphere Service Registry and Repository. The Configuration perspective allows an administrator to configure WebSphere Service Registry and Repository. This Configuration perspective might be the only available perspective if the Governance Enablement Profile has not been activated.

3. Navigate to **Manage Profiles** → **Configuration Profiles** → **Load Configuration Profile**.
4. Specify the location of the configuration profile compressed file and click **OK**.
5. Select **Governance Enablement Profile** and click **Make Active**. For the Initial Profile Status, the status of the newly loaded profile is set to “Archived”.

Registering the Web Services Description Language

To register a new service, you load the Web Services Description Language (WSDL) that defines a service and associate it with a service version that is used to govern that service. Create a new service version called Orders Service and use it to initiate the service life cycle.

To load the Orders Service into WebSphere Service Registry and Repository, perform the following steps:

1. Log on to the WebSphere Service Registry and Repository Web user interface.
2. Go to **View** → **Service Documents** → **WSDL Documents**, as shown in Figure 4-4.



Figure 4-4 Loading documents

3. Click **Load Documents**. The Load Documents page is displayed, as shown in Figure 4-5.

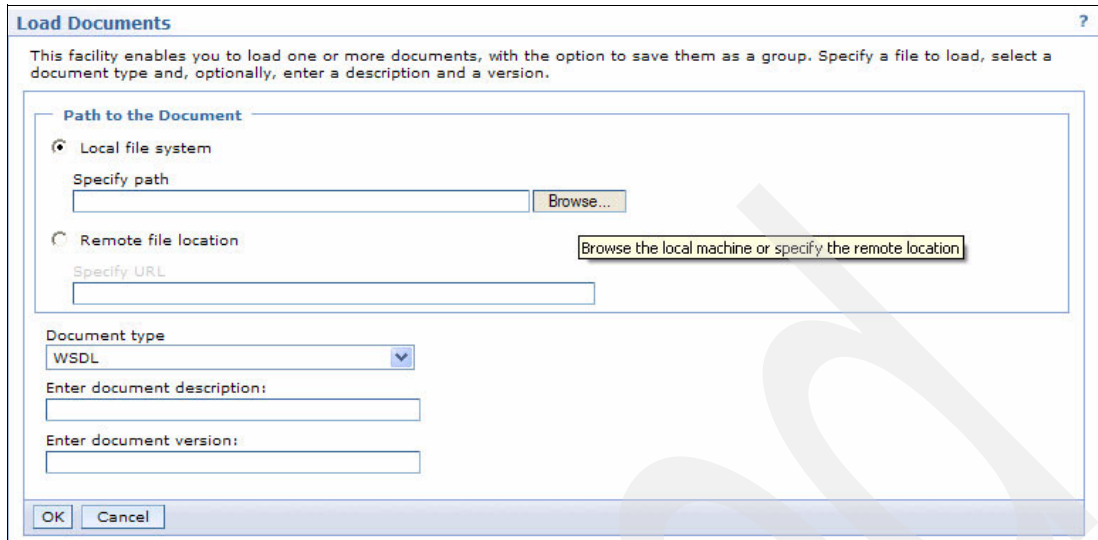


Figure 4-5 Loading the WSDL file

4. Click **Browse** to find the WSDL file that you want to load.
5. Enter the document description and version.
6. Click **OK**. A tree showing the WSDL and prerequisite files is displayed, as shown in Figure 4-6. To load the prerequisite files, click **Select Document**.

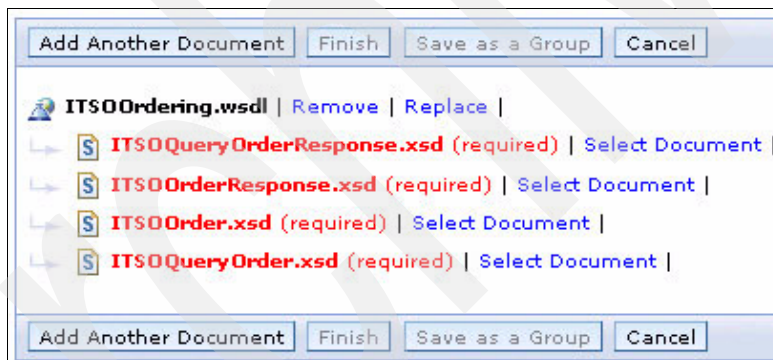


Figure 4-6 Dependent XSD documents of ITSOOrdering.wsdl

Dependent files: If a dependent document, such as an XSD file, is detected and identified as in a repository, it is automatically linked to the WSDL that is being loaded. You can override this process and load a new version, if you want.

7. Click **Finish** to load the WSDL document.

As the WSDL document is loaded, it is parsed to identify the WSDL logical objects, such as the Service, Port, Port type, and SOAP address.

Identifying the business capability through a business service

A *business capability* in the governance enablement profile is a construct that expresses a generalized capability within the SOA organization. Each business capability plays a particular role in the business processes of the organization and is therefore the starting point for traceability (from business to IT) in an SOA environment.

A business service represents a business capability that is viewed as a service within the organization and it passes through the capability life cycle, as shown in Figure 4-7.

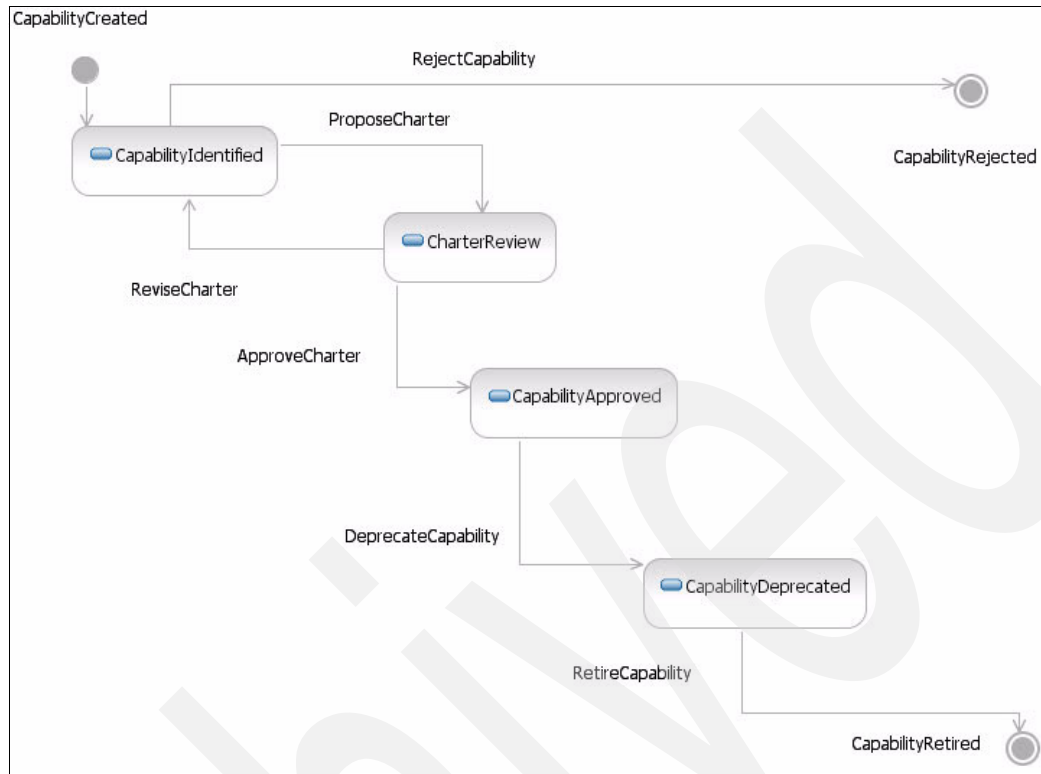


Figure 4-7 Capability life cycle diagram

Perform the following steps to create a new business service and enforce governance:

1. Log on to the WebSphere Service Registry and Repository Web user interface.
2. Switch to the Business perspective, if necessary, by selecting **Business** from the **Perspective** list.
3. Click **Task** → **Business Capability Tasks** → **Charter Business Capability**.
4. From the Identified Business Capabilities (Charter) list, click the **New Business Service**, as shown in Figure 4-8.

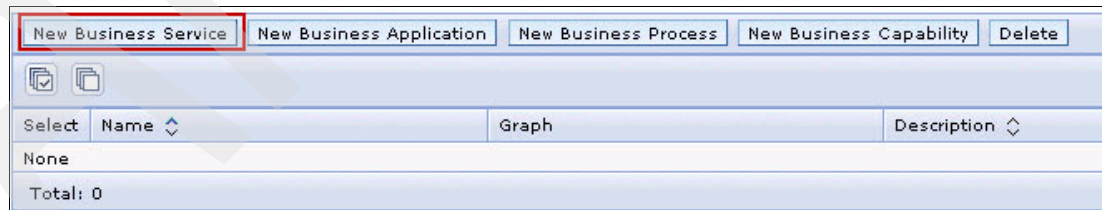


Figure 4-8 Creating New Business Service

5. Enter a Name and Description for the business service, as shown in Figure 4-9.

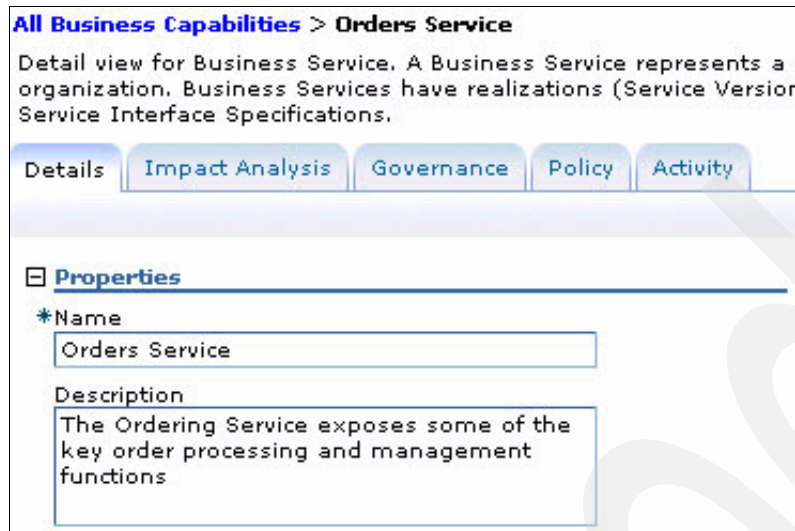


Figure 4-9 Business Service details

6. Click **Finish**.

Business Capability Identified: The Governance State of the new Business Service is Business Capability Identified, which is the initial state in the capability life cycle. A new Business Service is entered into this life cycle automatically.

7. Click **Edit Relationships**, as shown in Figure 4-10.

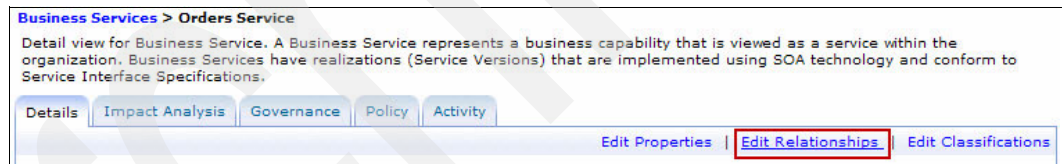


Figure 4-10 Edit Business Service Relationships

8. To the right of the Charter relationship, click **Add Other Document** and click **Load Document**, as shown Figure 4-11 on page 160.

Targets	
Orders Service Add Relationship	
↳ Versions	Add Capability Version
↳ Charter	Add Other Document
↳ Service Interface Versions	Add Service Interface Specification
↳ Owning Organization	Add Organization
↳ Artifacts	Add Document
↳ Dependency	Add Asset

Add Target

Use the options below to select a new target for the "Charter" relationship.

Search for Existing Entity

Name

Entity Type

Load Document

Document Type

Figure 4-11 Adding Charter document to Orders Service

Charter (Business Case) document: The *charter* is a separate document that describes the required capability in detail, including all functional and non-functional requirements. The charter is authored externally and then loaded into WebSphere Service Registry and Repository and attached to the business service. For a user to perform the Propose charter transition, a charter must be associated with the capability, as a target of the Charter relationship.

9. Click **Finish** to load the charter document. The charter document is added as a target of the Charter relationship.
10. To transition the service to the Charter Review state, click **Propose for Charter Review** under the Additional Properties of the Details tab.

Charter Review: The Charter Review state indicates that the service is ready for review and makes it available to the reviewers.

11. To the right of the Owning Organization relationship, click **Add Organization**.
12. Click asterisk (*) and select **Order Management** from the auto suggest list, as shown in Figure 4-12 on page 161. Click **Add**.

Use the options below to select a new target for the "Owning Organization" relationship.

Search for Existing Entity

Name

*

Commercial	1 result
Common services	1 result
Customer Relationship Management	1 result
ITSO Enterprises	1 result
Order Management	1 result

Figure 4-12 Search for an existing organization as a target owning organization

13. Click **Finish** to save your changes.
14. To transition the Business Service to the Business Capability Approved state, click **Approve Capability**. The new governance state is Business Capability Approved, as shown in Figure 4-13.

Governance State

Business Capability Approved

Figure 4-13 Business Capability Approved governance state

Approval Criteria: Before the Business Service can be transitioned to the approved state in the life cycle, make sure that the proposed capability does not duplicate other services in the registry and that an owning organization is assigned that will be responsible for all versions of this capability and for managing the requirements for this service.

Realizing the business service with a service version

A *service version* in the governance enablement profile represents a specific version, or release, of a service, and provides a range of functional and non-functional specifications that hold for that version of the service. Over time, multiple versions of the same business capability might be created, as the service is enhanced and extended. The service version is the representation of the realization of the business capability.

The capabilities of the service version are defined as service-level definitions (SLDs). In the case of a composite service, the services on which the service version depends can be identified through SLAs that are associated with the service-level definitions that are provided by the consumed service.

A service version passes through the SOA life cycle, as shown in Figure 4-14 on page 162.

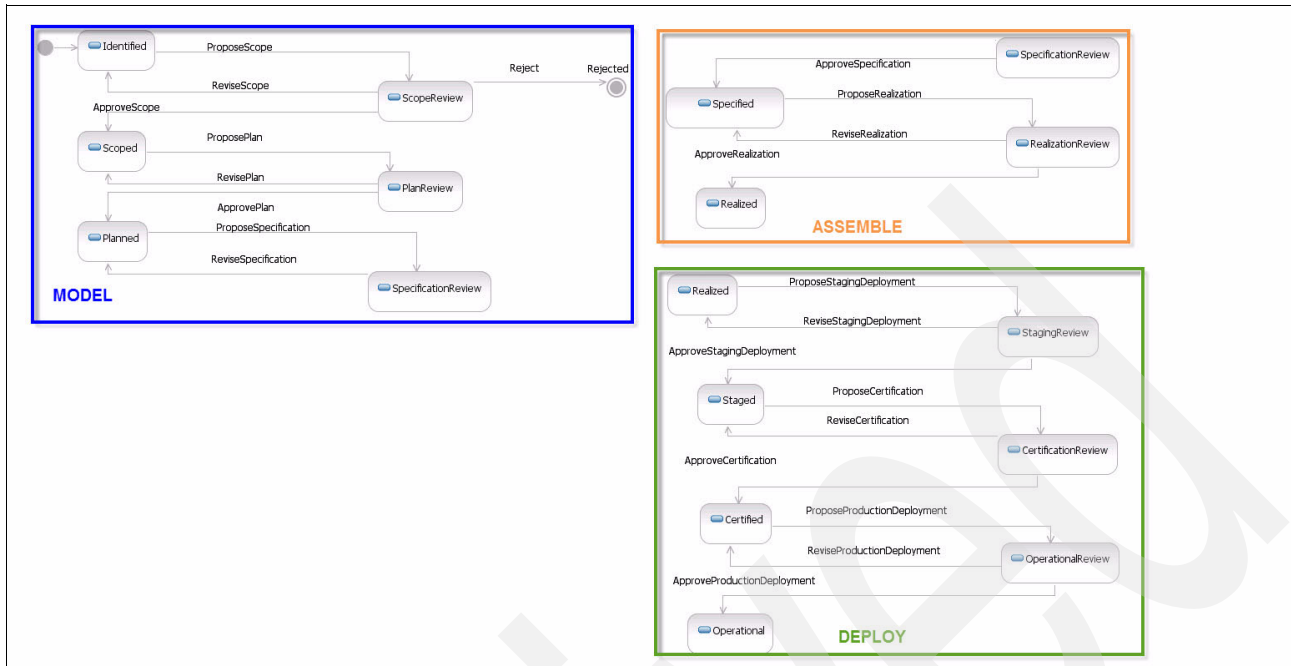


Figure 4-14 SOA life cycle

Perform the following steps to realize the Orders Service business service with a new service version, which corresponds to the imported WSDL:

1. Log on to the WebSphere Service Registry and Repository Web user interface.
2. Click **View** → **Business Capability Life Cycle** → **Approved Business Capabilities**, as shown in Figure 4-15.

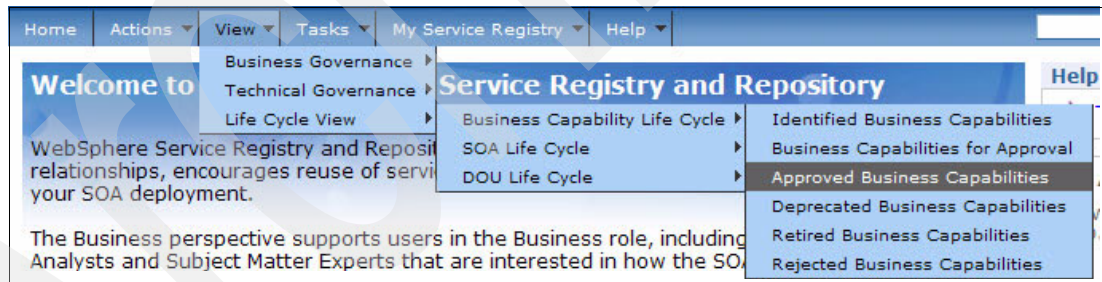


Figure 4-15 Viewing Approved Business Capabilities

3. Click **Edit Relationships**.
4. To the right of the Versions relationship, click **Add Capability Version**, as shown in Figure 4-16 on page 163.

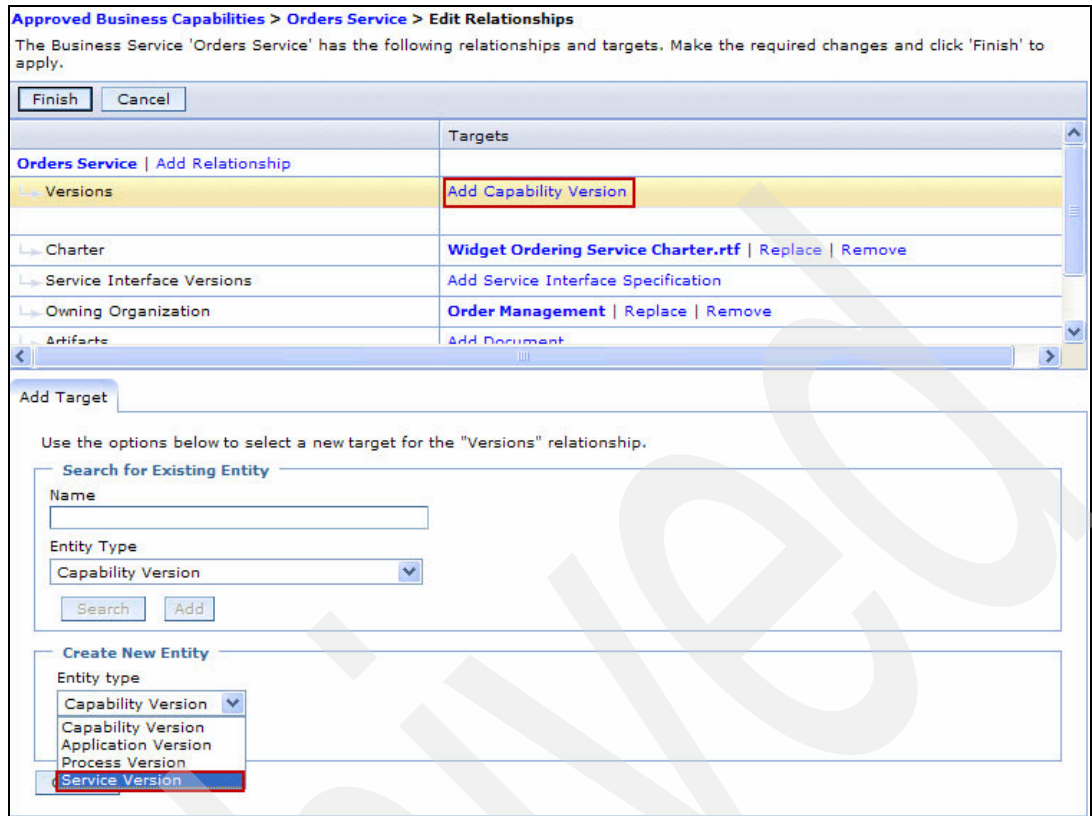


Figure 4-16 Adding Service Version

5. In the Create New Entity section, select **Service Version** from the Entity type drop-down list and click **Create**.
6. Enter a Name, Description, and Version for the service version, as shown in Figure 4-17.

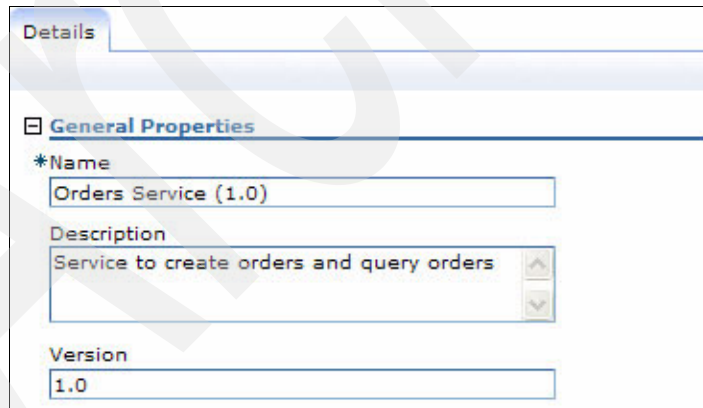


Figure 4-17 Service version details

7. To the right of the Provided Web Services relationship, click **Add Service** and select **ITSOOrdering**, as shown in Figure 4-18 on page 164.

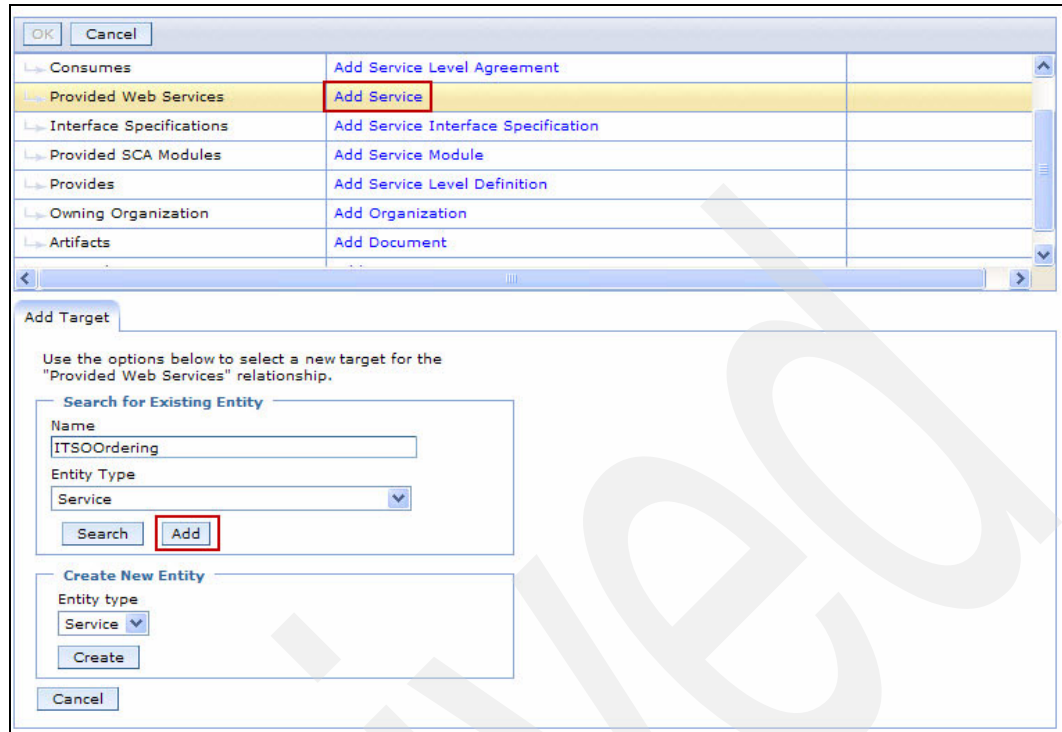


Figure 4-18 Updating the Provided Web Services relationship

Service entity: ITSOOrdering is the service entity that is derived from the imported WSDL file.

8. To assign an owning organization to the service version, to the right of the Owning Organization relationship, click **Add Organization**.
9. Click asterisk (*), select **Order Management** from the auto suggest list, and click **Add**.
10. Click **Finish** to save your changes. The service version is created and added as the target of the Capability Version relationship of the business version.

The governance state of our service version is *Identified*. This state is the initial state in the model phase of the SOA life cycle; a new service version is entered into the SOA life cycle automatically.

11. Click **Propose Scope** to transition the service version to the Scope Review state.

Policy assertions for proposing scope: For a user to perform the Propose scope transition on a service version, the following conditions must exist:

- ▶ The service version must be associated with a business capability.
- ▶ The description, version, and requirements properties must all have a value.

You get an error message in the case of a nonconformance to the policy, as shown in Figure 4-19 on page 165.

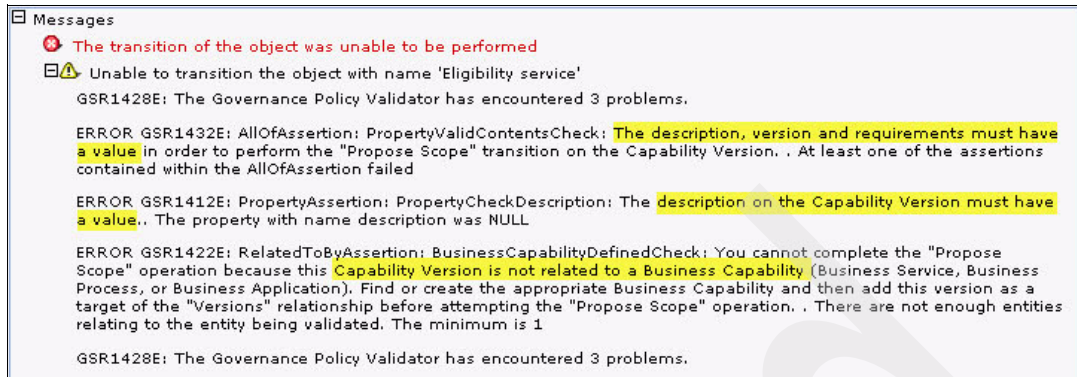


Figure 4-19 Governance Policy Validator

Reviewing the scope: Ensure that the service version passes through the following verification activities successfully:

- ▶ The service version is warranted throughout the organization.
- ▶ The requirements and stakeholders are in agreement.
- ▶ The owning organization that is responsible for delivering the requirements is identified and assigned to the service version.

12. Click **Approve Scope** to transition the service version to the Scoped state.
13. Click **Propose Plan** to transition the service version to the Plan Review state.
14. Click **Approve Plan** to transition the service version to the Planned state.
15. Click **Propose Specification** to transition the service version to the Specification Review state.

Policy assertions for proposing specification: For a user to perform the Propose specification transition on a service version, the following conditions must exist:

- ▶ Any SLA that is associated with the capability version must be in either an Active or Inactive state.
- ▶ The service version must reference a provided web service, a provided Service Component Architecture (SCA) module, or an interface specification.
- ▶ One of the following conditions must be true:
 - Any referenced service-level definitions must be in the SLD Subscribable state.
 - Either a module or a WSDL document must be assigned as the target of the Artifacts relationship.

16. Click **Approve Specification** to transition the service version to the Specified state.
17. Click **Propose Realization** to transition the service version to the Realization Review state, as shown in Figure 4-20.

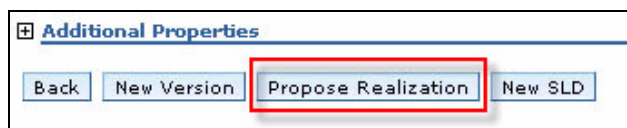


Figure 4-20 Proposing Realization

- Under Additional Properties, click **Approve Realization** to transition the service version to the Realized state, as shown in Figure 4-21.

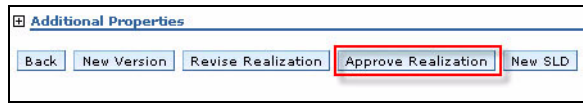


Figure 4-21 Approving Realization

The next state in the SOA life cycle is the Staging Review state. To transition the service version to the Staging Review state, you need to Propose Staging Deployment. The service version must have at least one Subscribable SLD with at least one Staging endpoint attached.

Creating an SLD for the service version

The SLD provides a formal specification of the physical communication mechanisms that are used to deliver the messages for interaction with a provided service. The SLD includes non-functional characteristics that are related to the interaction, such as security and identity.

To define the SLD to which the service version will adhere, perform the following steps:

- Log on to the WebSphere Service Registry and Repository web user interface.
- On the Home page under **Technical Governance**, click **Service Versions**, as shown in Figure 4-22.

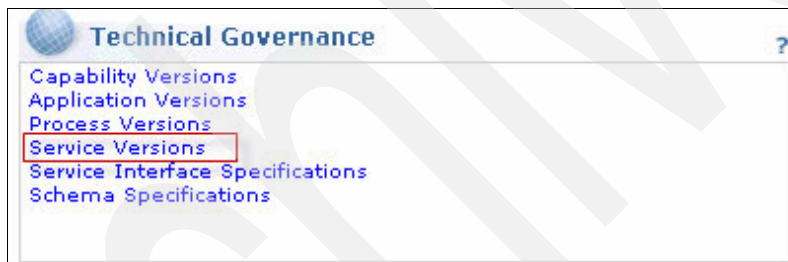


Figure 4-22 Service Versions

- Click **Orders Service (1.0)**.
- Under Additional Properties, click **New SLD**, as shown in Figure 4-23.

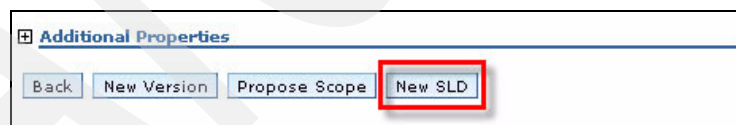


Figure 4-23 Adding a New SLD to a service version

- Click **SLD - Orders Service (1.0)** under **Provides** to display the SLD details.
- Click **Edit Properties** and change the Name to **SLD - Orders Service (1.0) - Ordering**.

Service interface relationship cardinality: Service Interface relationship cardinality is 0,1. Ordering Service provides two service interfaces (ITSOOrderQuery and ITSOOrdering), so define another SLD that corresponds to the other service interface.

- For Average Response Time, type 100. For Availability, select **24/7 High Availability** from the list box, as shown in Figure 4-24 on page 167.

Figure 4-24 Editing the SLD properties

Quality of service (QoS) of the SLD: The extended service-level definition in the governance enablement profile provides additional properties that are related to QoS information:

- ▶ Average response time: The typical response time, in milliseconds, in an interaction with this endpoint (or service-level definition)
- ▶ Availability: The level of availability that consumers can expect from this endpoint

You can also add additional properties, as shown in Figure 4-25 on page 168.

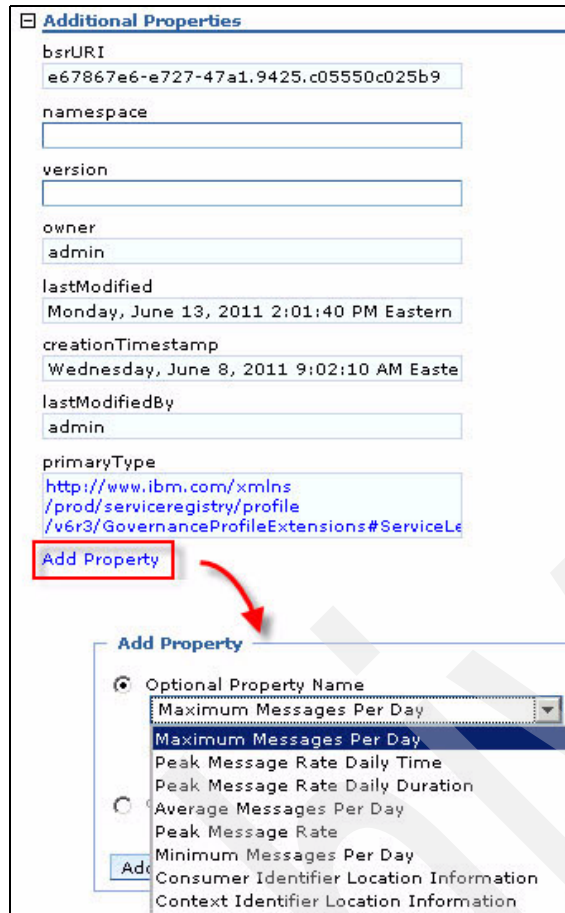


Figure 4-25 Adding additional QoS information to the SLD

8. Click **OK** to save your changes.
9. Click **Edit Relationships**.
10. To the right of the Available Endpoints relationship, click **Add Service Endpoint**.
11. Click the asterisk (*), select the required service endpoint from the auto suggest list, and click **Add**, as shown in Figure 4-26 on page 169.

Service endpoint: The *service endpoint* is the derived logical representation of the endpoint within the imported WSDL document. The WSDL that we loaded previously contained the production endpoint. By associating the production endpoint with the SLD, the service becomes consumable in the production environment.

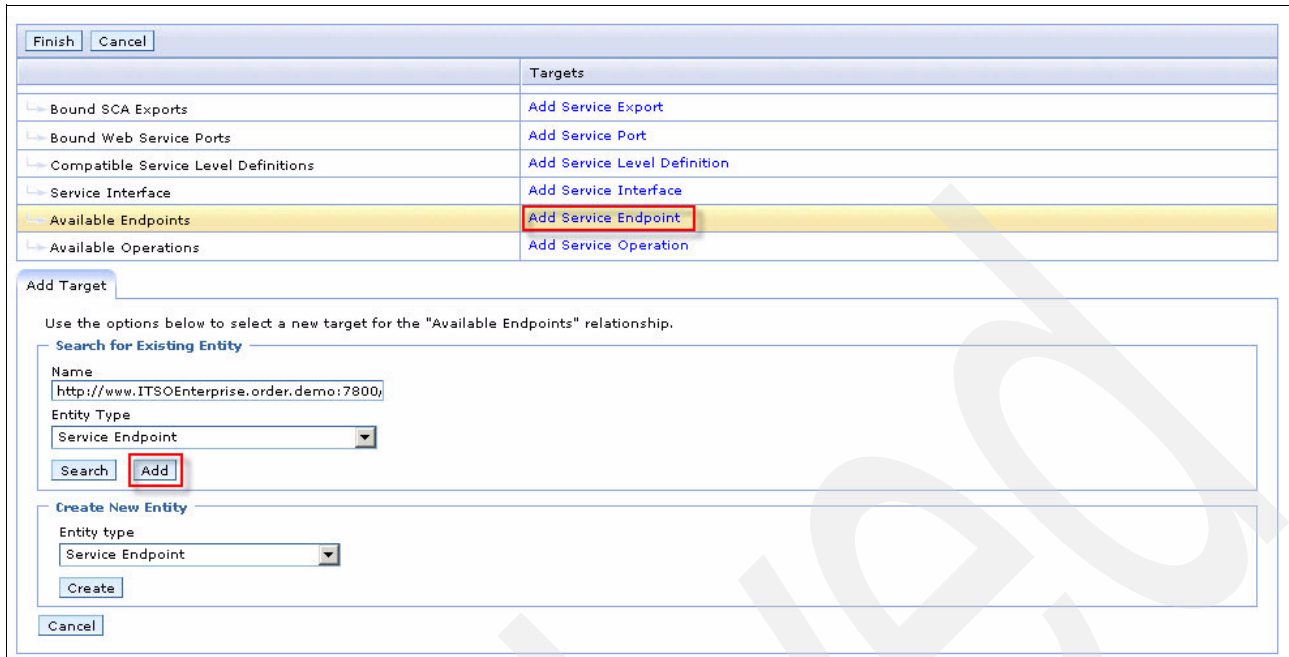


Figure 4-26 Adding service endpoint that realizes the SLD

12. To the right of the Service Interface relationship, click **Add Service Interface**.
13. Click the asterisk (*), select **ITSOOrdering** from the auto suggest list, and click **Add**.

Service interface: The service interface is the derived logical representation of the *port type* within the imported WSDL document.

14. To the right of Bound Web Service Ports, click **Add Service Port**.
15. Click the asterisk (*), select **ITSOOrderingSOAP** from the auto suggest list, and click **Add**.

Service port: The service port is the derived logical representation of the *port* within the imported WSDL document.

16. Click **Finish**, as shown in Figure 4-27.



Figure 4-27 Finalizing the SLD relationships update

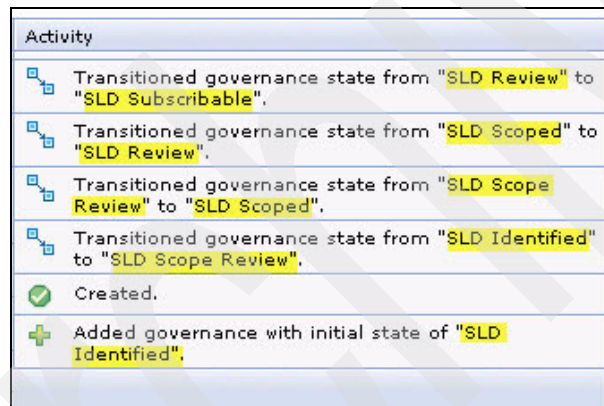
SLD Identified: For an SLD to become active, the associated SLD must be in a Subscribable state. The Governance State of the new SLD is Business Capability Identified, which is the initial state in the capability life cycle; a new Business Service is entered into this life cycle automatically.

17. To make the SLD active, complete the following steps:

- a. To transition the SLD to the Scope Review state, click **Propose Scope** under the Additional Properties of the Details tab. The Governance State is now Scope Review.
- b. Click **Approve Scope**. The Governance State becomes Scoped.
- c. Click **Propose Specification**. The Governance State is now Review.
- d. Click **Approve Specification**. The Governance State becomes Subscribable, and the SLD is now ready to use.

SLD subscribable: The SLD is *subscribable*, which means that SLAs can now be requested against it.

18. From the Activity tab, you can access the activity audit information, which displays all the transition steps, as depicted in Figure 4-28.







Activity	
	Transitioned governance state from "SLD Review" to "SLD Subscribable".
	Transitioned governance state from "SLD Scoped" to "SLD Review".
	Transitioned governance state from "SLD Scope Review" to "SLD Scoped".
	Transitioned governance state from "SLD Identified" to "SLD Scope Review".
	Created.
	Added governance with initial state of "SLD Identified".

Figure 4-28 Viewing the life cycle history

Activating the service endpoint and completing the service life cycle

After we have described the service in sufficient detail for it to be consumed and programmed against, and have deployed an implementation of the service, we must declare the endpoint in the registry, so that it is promoted to, and therefore visible in, the appropriate environments.

To complete the services life cycle of the Ordering service version, perform the following steps:

1. Click **Tasks** → **Endpoints Tasks** → **Endpoints For Activation**.
2. From the **Offline Endpoints** list, select the endpoint.
3. Click **Edit Classifications**, as shown in Figure 4-29 on page 171.

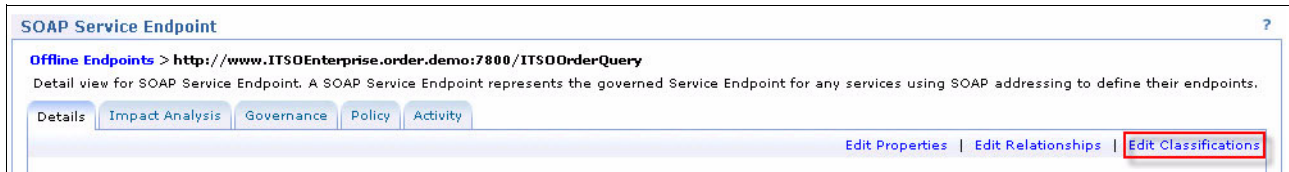


Figure 4-29 Edit Classifications

4. Expand **Governance Profile Taxonomy** → **Environment**.
5. In the Classifications tree, select **Staging** and click **Add**. The Staging classification is added to the Classification list, as shown in Figure 4-30.

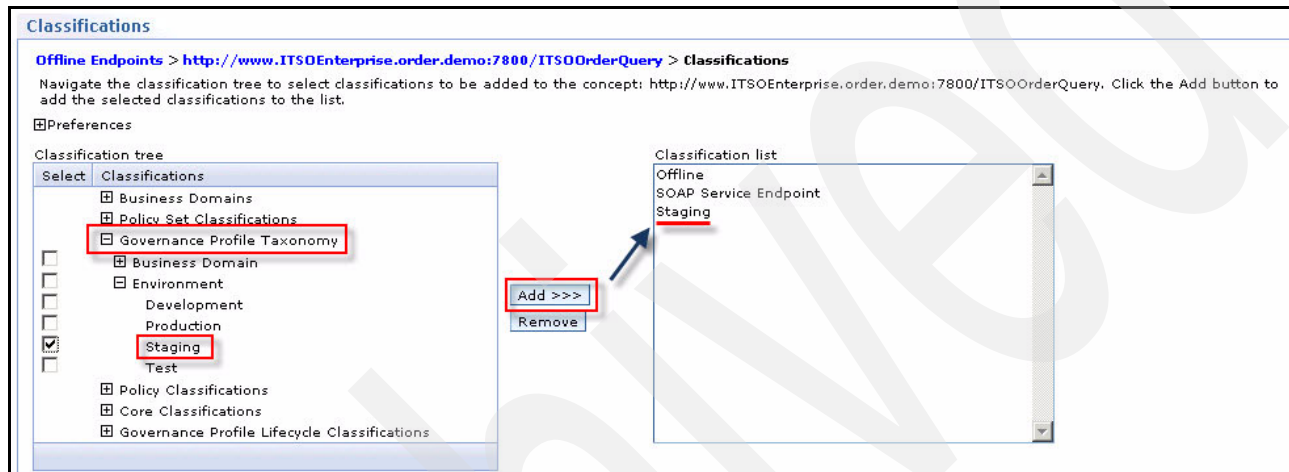


Figure 4-30 Adding the Staging classification to the service endpoint

6. Click **OK** to assign the classification.
7. Click **Approve For Use**, and note that the endpoint governance state changes to Online, as shown in Figure 4-31.

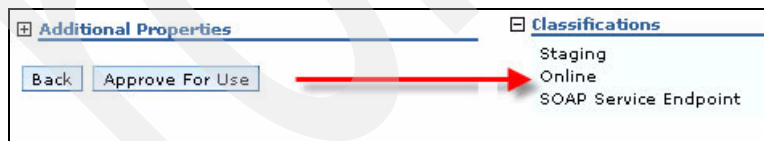


Figure 4-31 Approving the endpoint for use

8. Under **Technical Governance**, click **Service Versions**.
9. Click **Orders Service (1.0)**.
10. Click **Propose Staging Deployment** to transition the service version to the Staging Review state, as shown in Figure 4-32.

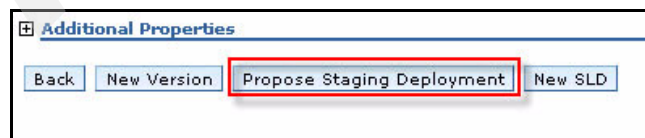


Figure 4-32 Proposing Staging Deployment

Policy assertion for proposing staging deployment: For a user to perform the Propose staging deployment transition on a service version, there must be at least one subscribable SLD associated with the service version, which has an available endpoint with the Staging classification.

11. Click **Approve Staging Deployment** to transition the service version to the Staging review state.

Staging environment: In the staging environment, testing the service is performed to ensure that, if the service is made available in the production environment, it will meet all of its proposed SLDs.

12. Click **Propose Certification**. The governance state becomes Certification Review.
13. Click **Approve Certification** to transition the service version to the Certified state.
14. To promote the service version to the production state, add the **Production** classification to the service endpoint as described in step 3 on page 170 and the subsequent three steps.
15. Click **Propose Production Deployment** to transition the service version to the Operational Review state.

Policy assertion for proposing production deployment: For a user to perform the Propose production deployment transition on a service version, there must be at least one subscribable SLD, associated with the service version, that has an available endpoint with the Production classification.

16. Click **Approve Production Deployment**, and note the new governance state is Operational and that a Deprecate is now displayed. *Deprecation* is the next stage in the life cycle when the service is to be phased out.

The new Service Version of the Ordering service is now available in the production environment.

Registering and governing the service consumer

For this scenario, the consumer organization is Customer Relationship Management. We use a service version, because the consumer also provides a service, so that the CSRQueryOrder service is available to other consumers within the Customer Relationship Management organization. It has its own service, with a separate interface, and its own endpoints, and it relies on calling the Orders service to complete its processing. So, we model the CSRQueryOrder consumer service in the same way that we modeled the Orders provider service. Refer to the following steps to register and govern the service:

- ▶ “Registering the Web Services Description Language” on page 156
- ▶ “Identifying the business capability through a business service” on page 157
- ▶ “Realizing the business service with a service version” on page 161
- ▶ “Creating an SLD for the service version” on page 166
- ▶ “Activating the service endpoint and completing the service life cycle” on page 170

Creating an SLA between the service consumer and the service provider

The SLA defines a dependency between a service version and the SLD representing the service provider. In this scenario, the SLA is between the consumer, CSR, and the provider, Orders Query Service.

Perform the following steps to define the SLA:

1. Log on to the WebSphere Service Registry and Repository Web user interface.
2. On the Home page under **Technical Governance**, click **Service Versions**.
3. Click **CSRQueryOrder Service(1.0)**.
4. Assign **Consumer Identifier** to the CSR Order Query Service Version, as shown in Figure 4-33.

The screenshot shows the 'Service Versions' page for 'CSRQueryOrder Service (1.0)'. The page has a breadcrumb trail 'Service Versions > CSRQueryOrder Service (1.0)'. Below the breadcrumb is a description: 'Detail view for Service Version. A Service Version represents a specific service. The Service Version exposes its capabilities as service level definitions provided by the consumed service.' There are five tabs: 'Details', 'Impact Analysis', 'Governance', 'Policy', and 'Activity'. The 'Details' tab is selected. Under the 'Properties' section, the following fields are visible: '*Name' (CSRQueryOrder Service (1.0)), 'Description' (empty), 'Version' (1.0), 'Consumer Identifier' (OrderQueryConsumerCSRAccount, highlighted with a red box), 'Version Availability Date' (Monday, June 13, 2011), 'Version Termination Date' (Monday, June 13, 2011), 'Version Requirements Link' (urn:serviceregistry), 'Asset Web Link' (urn:serviceregistry), and 'Remote State' (empty).

Figure 4-33 Assigning Consumer Identifier to consumer service version

Consumer Identifier: The Consumer Identifier provides a key that is used to identify the consuming capability version in any interaction where the version has subscribed to another provider/capability. During any interactions between the consumer and provider, the Consumer Identifier is expected to be included in the interaction header, which allows the provider ESB to identify the particular capability version that is invoking the endpoint.

5. Click **OK** to save your changes.
6. Click **Edit Relationships**.
7. To the right of the Consumes relationship, click **Add Service Level Agreement**, as shown in Figure 4-34 on page 174.

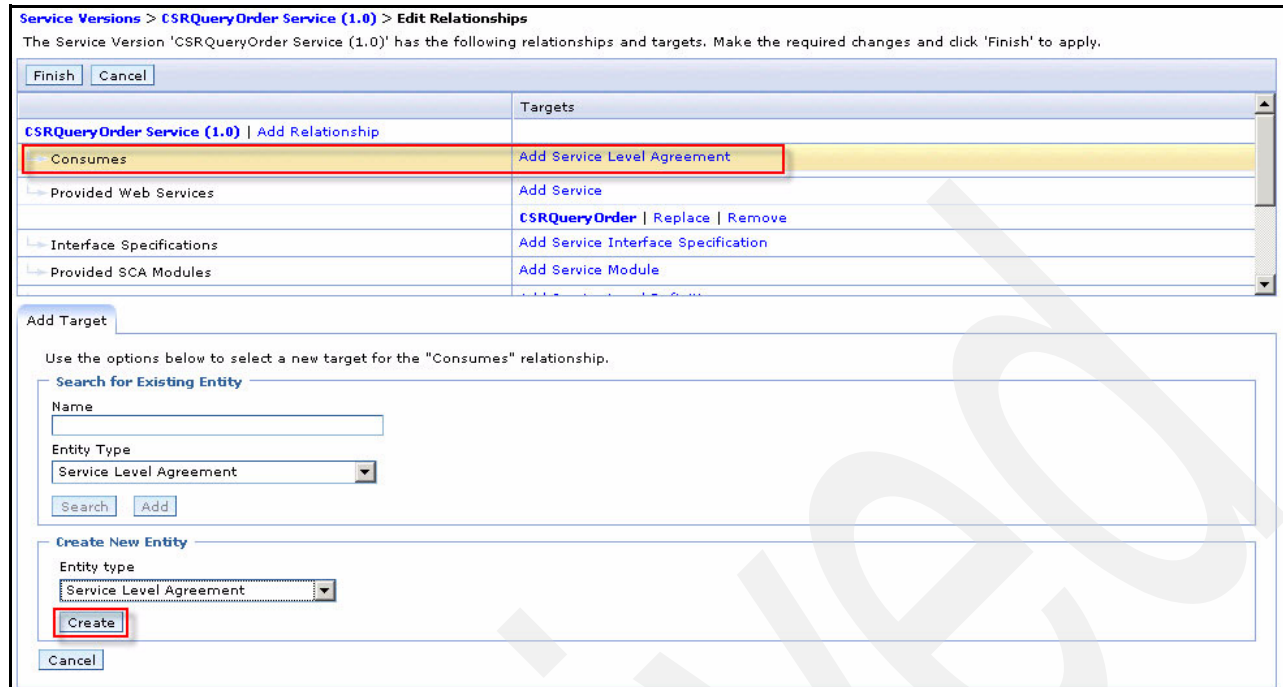


Figure 4-34 Adding the target SLA of Consumes relationship of the service version

8. Select the **Service Level Agreement** from the Entity type drop-down list in the Create New Entity box and click **Create**, as shown in Figure 4-34.
9. Give a **Name** to the SLA and specify a **Context Identifier**.

Context Identifier: This property identifies the specific service version that the consumer intends to consume. It allows a consumer to identify alternate SLAs that will be used when invoking the same provided service or endpoint (SLD). During any interactions between the consumer and provider, the Context Identifier is expected to be included in the interaction header (with the Consumer Identifier), which allows the provider ESB to identify the particular capability version context that is invoking the endpoint.

10. To the right of the Agreed Endpoints relationship, click **Add Service Level Definition**.
11. Click the asterisk (*), select **SLD - Orders Service (1.0) - Query** from the auto suggest list, and click **Add**. SLD - Orders Service (1.0) - Query is associated with the SLA, as a target of the Agreed Endpoints relationship, as shown in Figure 4-35.

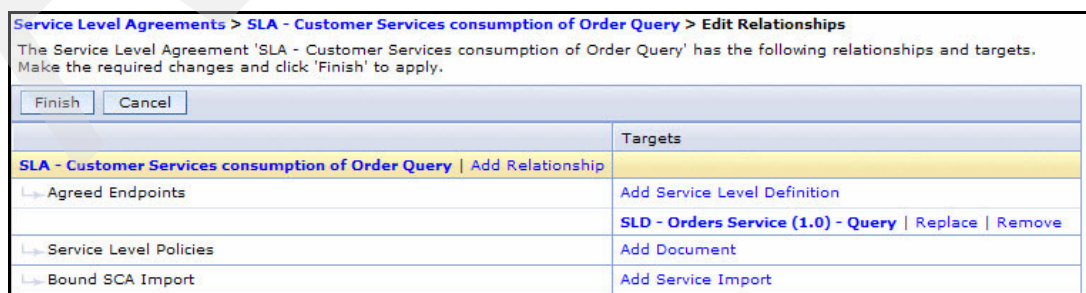


Figure 4-35 Agreed Endpoints relationship

Agreed Endpoints: The specific SLD in the providing capability that this SLA is intended to use. The SLD allows any endpoint addresses that realize that SLD to be located.

12. Click **Finish**. The Governance State of the SLA is now the Identified state.

Activating SLA: The SLA must move through a number of governance states to become active.

13. Click **Request SLA**. The Governance State of the SLA is now Requested.

14. Click **Approve SLA Request**. The Governance State of the SLA becomes Inactive.

15. Click **Activate SLA**. The Governance State of the SLA is now Active and it is ready to use, as shown in Figure 4-36.



Figure 4-36 SLA Governance State

4.3.3 IBM WebSphere ESB configuration to establish connection with WebSphere Service Registry and Repository

It is a requirement to create a WebSphere Service Registry and Repository definition in WebSphere ESB to establish a connection between IBM WebSphere ESB and WebSphere Service Registry and Repository. You can create, configure, and view all your WebSphere Service Registry and Repository access definitions using the IBM WebSphere ESB administrative console.

A WebSphere Service Registry and Repository definition and its connection properties are the mechanism that is used to connect to a registry instance and look up a Web service to invoke.

Each definition holds a set of properties that identify a WSRR instance and specify how you can access it. To create a new WebSphere Service Registry and Repository definition, use the administrative console to perform the following steps:

1. As an administrator user, log on to the **Integrated Solution Console** of WebSphere ESB.
2. Expand **Service integration** → **WSRR definitions** in the navigation pane. The WSRR definitions page is displayed in the content pane, as shown in Figure 4-37 on page 176.

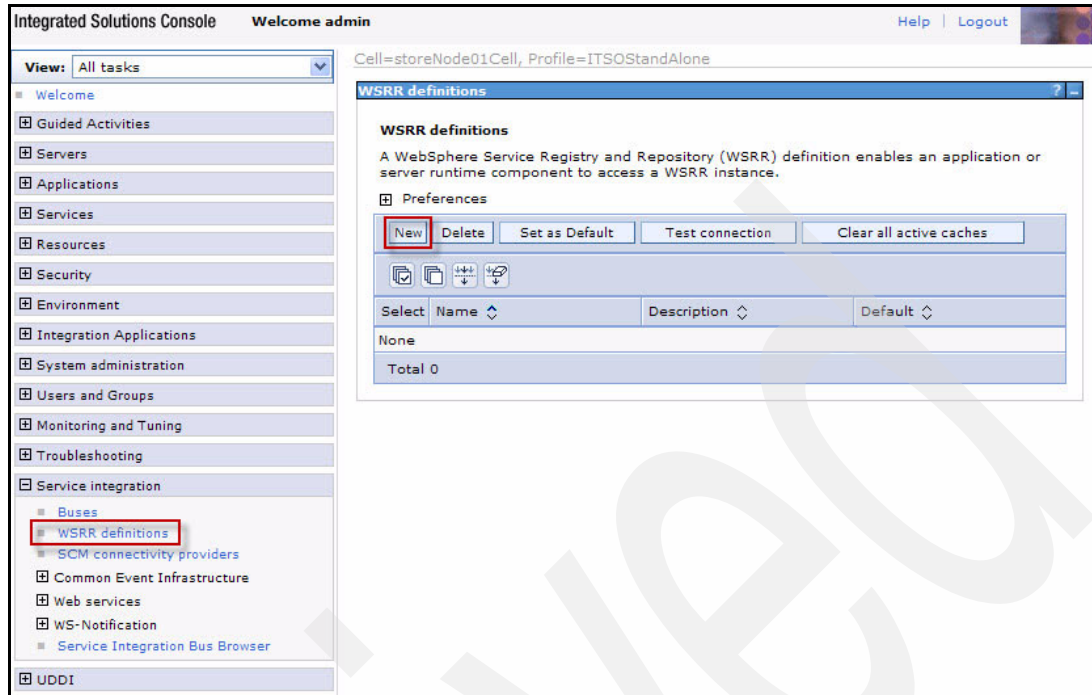


Figure 4-37 WSRR definitions

3. Click **New**. The WSRR definitions Configuration page displays. Enter the details, as shown in Figure 4-38.

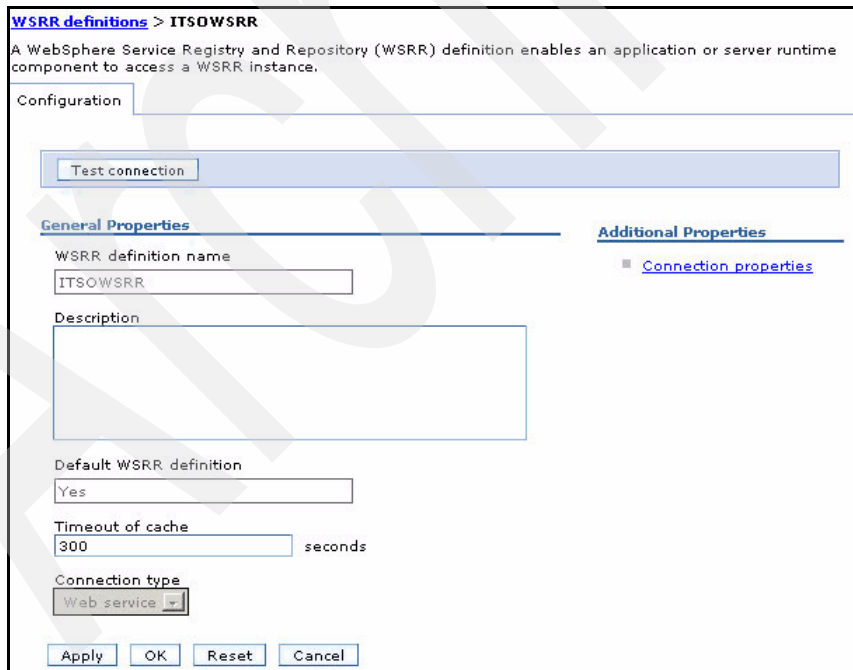


Figure 4-38 WSRR definition Configuration page

Timeout of cache: The time (in seconds) after which the results in the WSRR cache expire and will be refreshed. This value can be changed from the default value. If you specify a value of 0 (zero), results are never cached. The cache is used for storing information about service endpoints and mediation policies, which is retrieved from the registry.

4. Click **Apply** to save these properties.
5. In the content pane, under Additional Properties, click **Connection properties**. The Connection properties Configuration page displays, as shown in Figure 4-39.

[WSRR definitions](#) > [ITSOWSRR](#) > [Web service](#)
Connection properties for this WSRR definition.

Configuration

General Properties

Connection type
Web service

* Registry URL
https://9.12.5.225:9443/WSRRCoreSDO/services/WSRRCoreSDOPort

Authentication alias
storeNode01/WSRRdefinition

SSL Configuration
NodeDefaultSSLSettings

Apply OK Reset Cancel

Related Items

- JAAS - J2C authentication data
- SSL Configurations

Figure 4-39 WSRR definition connection properties

Registry URL: You need to follow steps 6 through 7 in case WebSphere Service Registry and Repository runs as a secure Java 2 Platform, Enterprise Edition (J2EE) application on WebSphere Application Server. The Registry URL is used to connect to the WSRR instance. The default value is `http://localhost:9080/WSRRCoreSDO/services/WSRRCoreSDOPort`.

6. Select an authentication alias to authenticate with the WSRR instance from the Authentication alias drop-down menu. To define a new authentication alias, under Related Items, click **JAAS - J2C authentication data**. Click **New** on the JAAS - J2C authentication data page, as shown in Figure 4-40 on page 178.

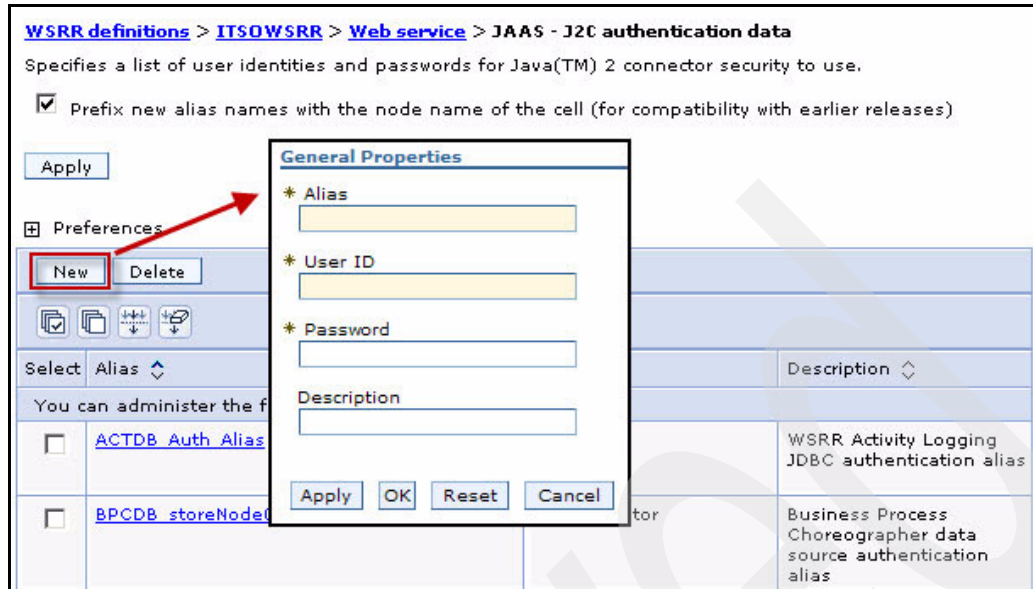


Figure 4-40 Create a new authentication alias

7. On the Web service page, select **NodeDefaultSSLSettings** from the SSL Configuration drop-down menu, as shown in Figure 4-41.

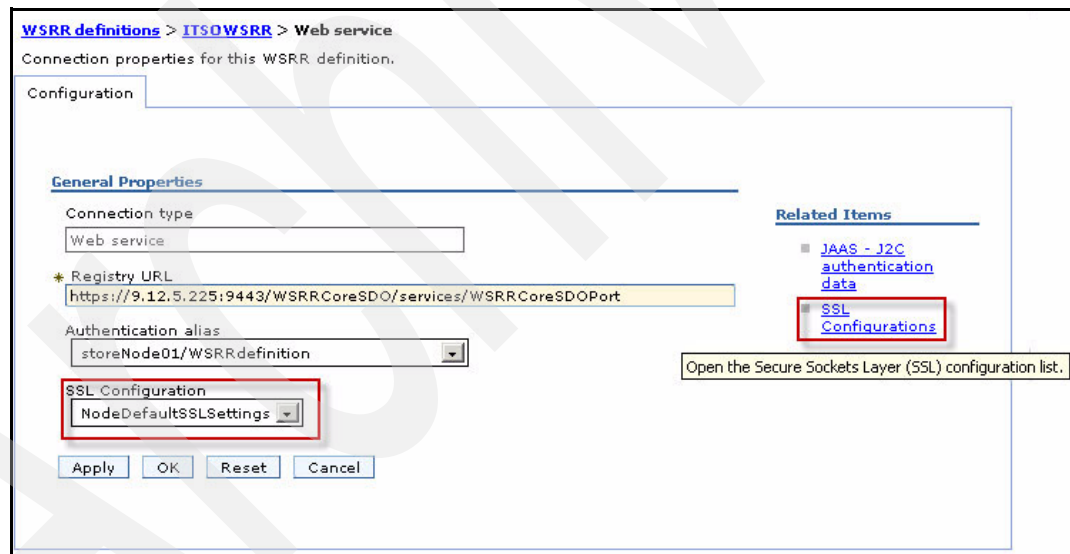


Figure 4-41 SSL configuration

SSL Configuration: For this field, type the Secure Sockets Layer (SSL) configuration to be used to communicate with a secured WSRR instance.

8. Click **Apply** to save these properties.
9. Click **Test Connection** to test the connection to the WSRR instance that has been defined. If successful, the following message is displayed, as shown in Figure 4-42 on page 179.

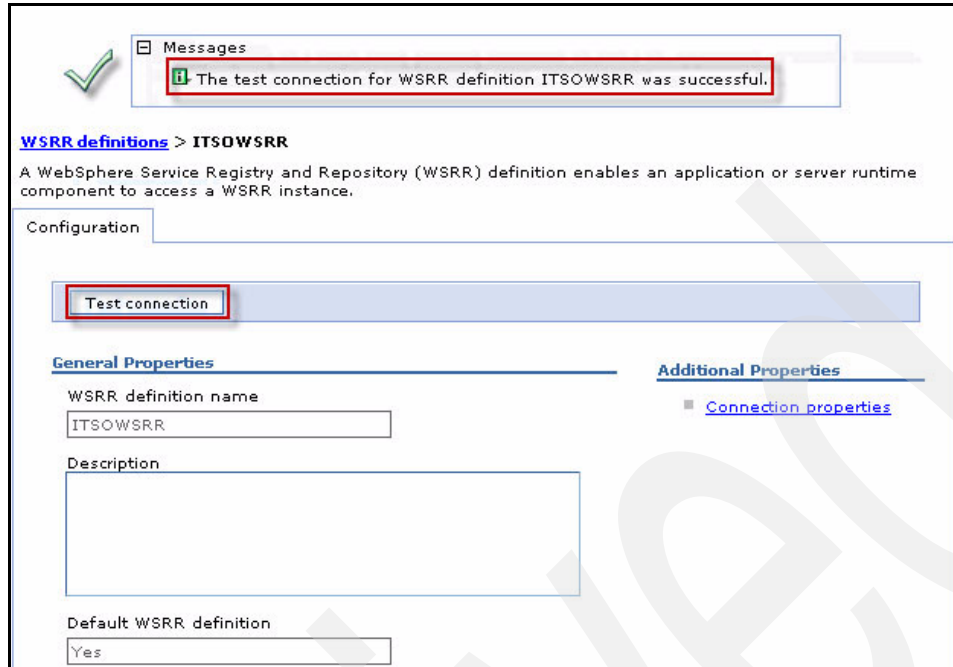


Figure 4-42 Testing connection to WSRR instance

10. Click **Save** to apply your changes to the master configuration and click **Save** again to complete the procedure.

Configuring WebSphere Service Registry and Repository connection in Integration Designer

To discover the web services and other artifacts, such as WSDL and schema files that are published to WebSphere Service Registry and Repository, Integration Designer requires a WebSphere Service Registry and Repository connection profile. To define a WebSphere Service Registry and Repository connection configuration, perform the following steps:

1. From the menu, select **Window** → **Preferences** to open the Preferences window.
2. Expand **Service Registries** and click **WebSphere Service Registry and Repository**, as shown in Figure 4-43 on page 180.

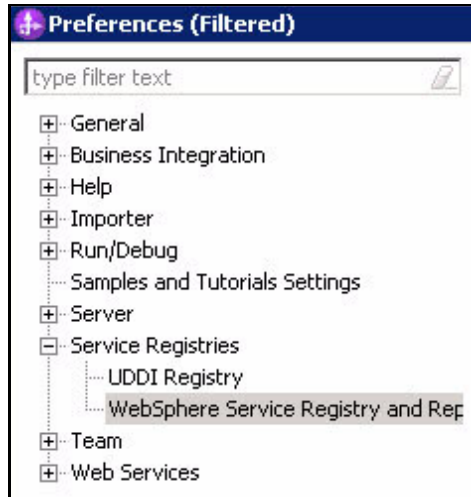


Figure 4-43 Service Registries category

3. Click **Add** to add a new definition.
4. Enter the connection details, as shown in Figure 4-44.

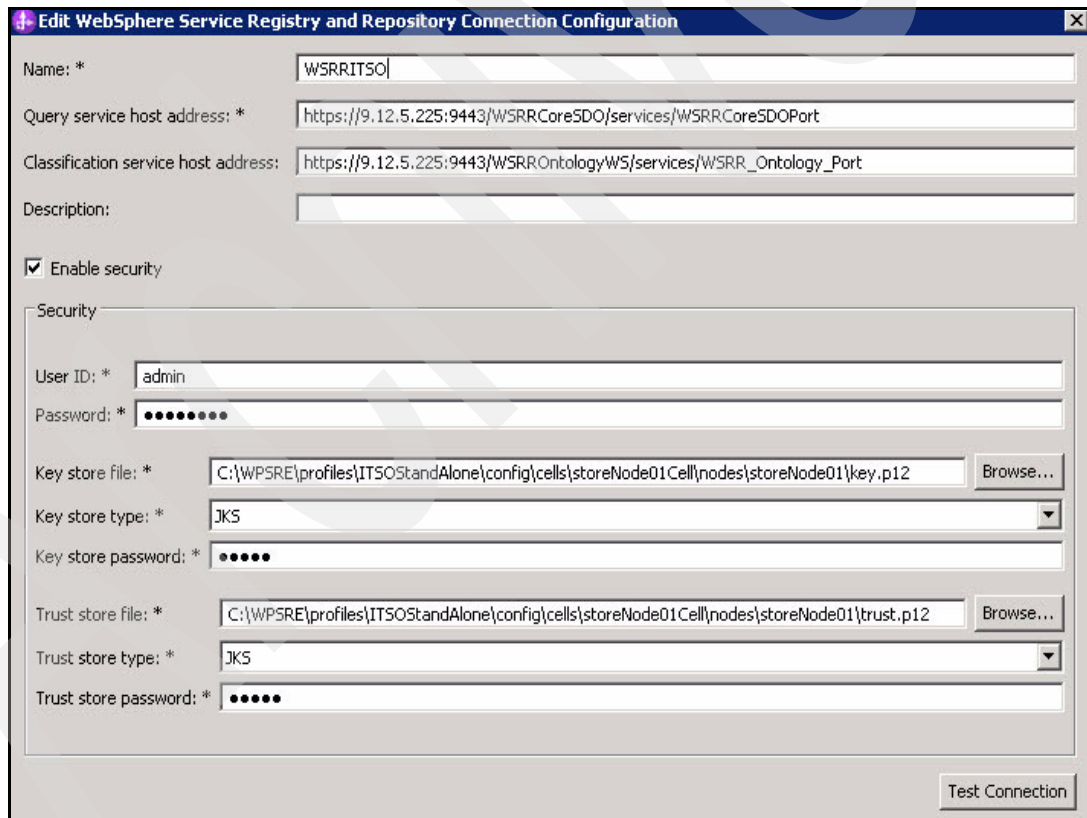


Figure 4-44 WebSphere Service Registry and Repository Connection Configuration

Enabling security: If your organization requires security to access the WebSphere Service Registry and Repository, select **Enable Security** on Figure 4-44 on page 180. You need to upload the keystore and truststore files, which are used by WebSphere Service Registry and Repository, and provide the type and password information.

5. Click **Test Connection** to see if you can connect to the registry.
6. If the connection is successful, click **OK** and you are returned to the Preferences window.
7. Click **OK** in the Preferences window to close the Preferences window.

4.3.4 Developing the mediation module

To implement the scenario in WebSphere ESB, use following steps:

1. Import the ITSO Ordering Library into the IBM Integration Designer workspace.
2. Build the mediation module.
3. Assemble the mediation module.
4. Develop the mediation flow.
5. Test the mediation module.

Importing the ITSO Ordering Library into the IBM Integration Designer workspace

The ITSO Ordering Library contains all artifacts, such as business objects, interfaces, and WSDL files, that relate to the query ordering service.

From the Integration Designer top-level menu, follow these steps:

1. Select **File** → **Import**.
2. In the Import dialog box, select the **Project Interchange** format from the other folder.
3. On the Import Projects window, select the library and click **Finish**, as shown in Figure 4-45 on page 182.

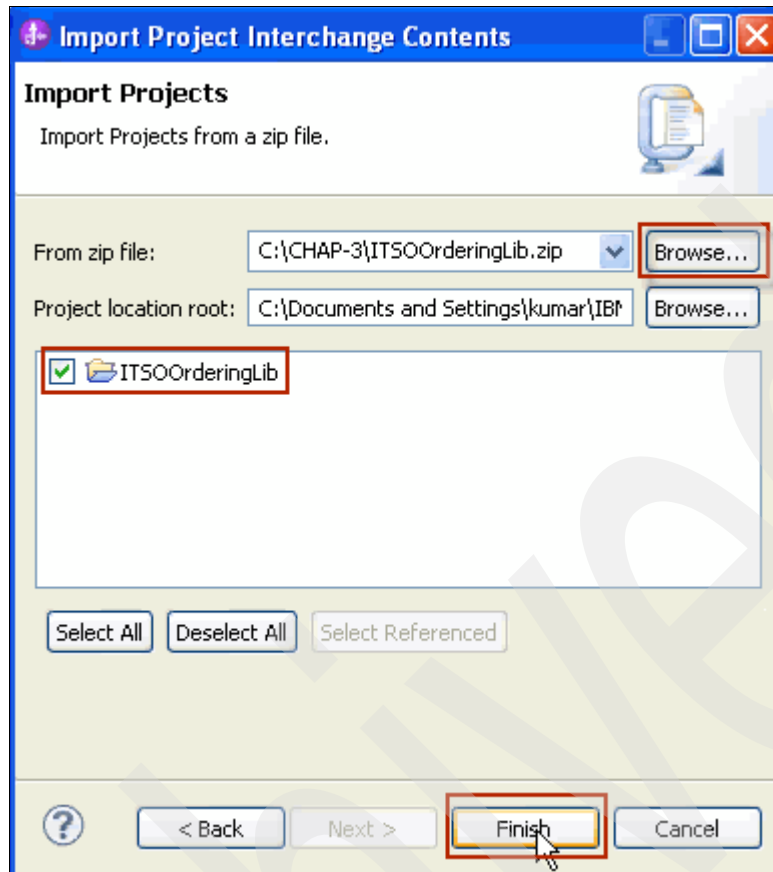


Figure 4-45 Import Projects interchange dialog

4. In the Business Integration view, by expanding the library project tree, you can see ITSOOrderingLib, as described in Figure 4-46 on page 183.

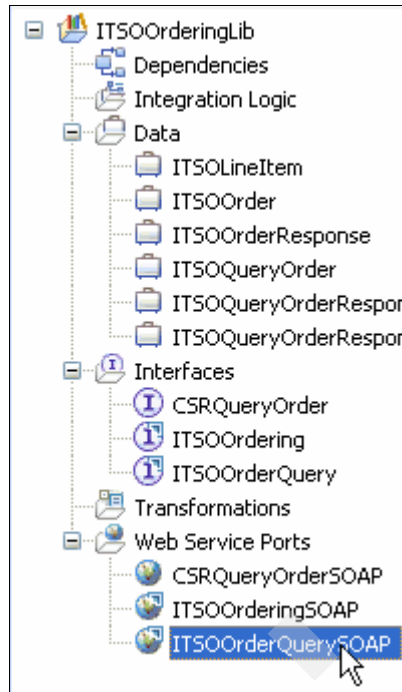


Figure 4-46 Library project view

Building the mediation module

We explain how to create the mediation module:

1. In the Business Integration view, right-click in the white space and select **New** → **Mediation Module**. For the module name, type **ITSOCSRSLACheck**.
2. On the Select the Required Libraries window, select **ITSOOrderingLib** as a referenced library, as shown in Figure 4-47.

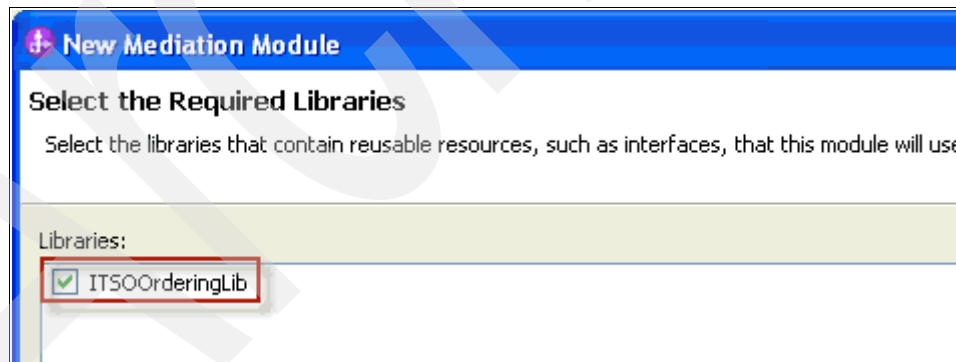


Figure 4-47 Library selection

3. On the next window, Select **Lazy parsing** and click **Finish**.
4. You will see the **ITSOCSRSLACheck** mediation component in the Assembly Diagram.

Assembling the mediation module

We add the CSR Query Order Interface and ITSO Order Query Interface into an Assembly Diagram. Our goal is to expose our mediation flow as a SOAP/HTTP web service. The CSR application can access this Web service to query the order by calling the ITSO QueryOrder Web service as a reference interface from the mediation module. Follow these steps:

1. In the Business Integration view, expand **ITSOOrderingLib** and select the **CSRQueryOrder** interface. Drag it onto the Assembly Diagram editor and release it, as described in Figure 4-48.

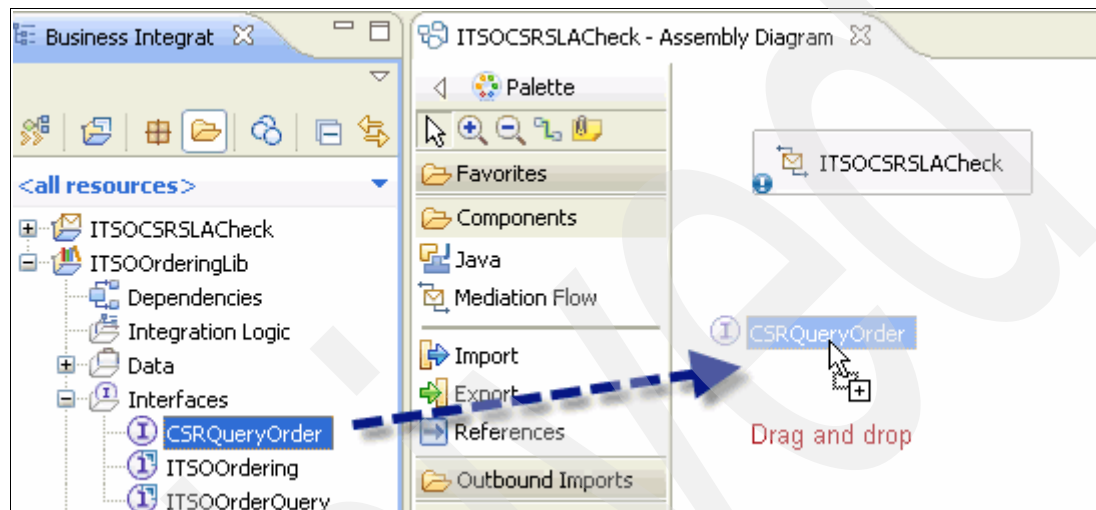


Figure 4-48 Drag the interface to the Assembly Diagram

2. In the Component Creation dialog box, as shown in Figure 4-49, select **Export with Web Service Binding** and click **OK**.

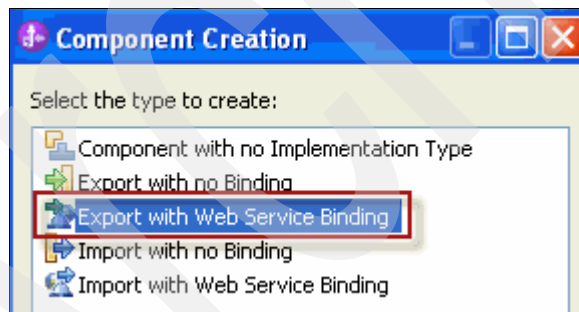


Figure 4-49 Component Creation

3. In the next window, for a transport protocol, select **SOAP1.1/HTTP** and click **Finish**.
4. You see that the WSDL file is generated in the Web service ports of the ITSOOrderingLib project. The Assembly Diagram is shown in Figure 4-50 on page 185.



Figure 4-50 Assembly Diagram

5. In the Business Integration view, expand **ITSOOrderingLib** and select the **ITSOOrderQuery** interface. Drag it onto the Assembly Diagram editor and release it, as described in Figure 4-51.

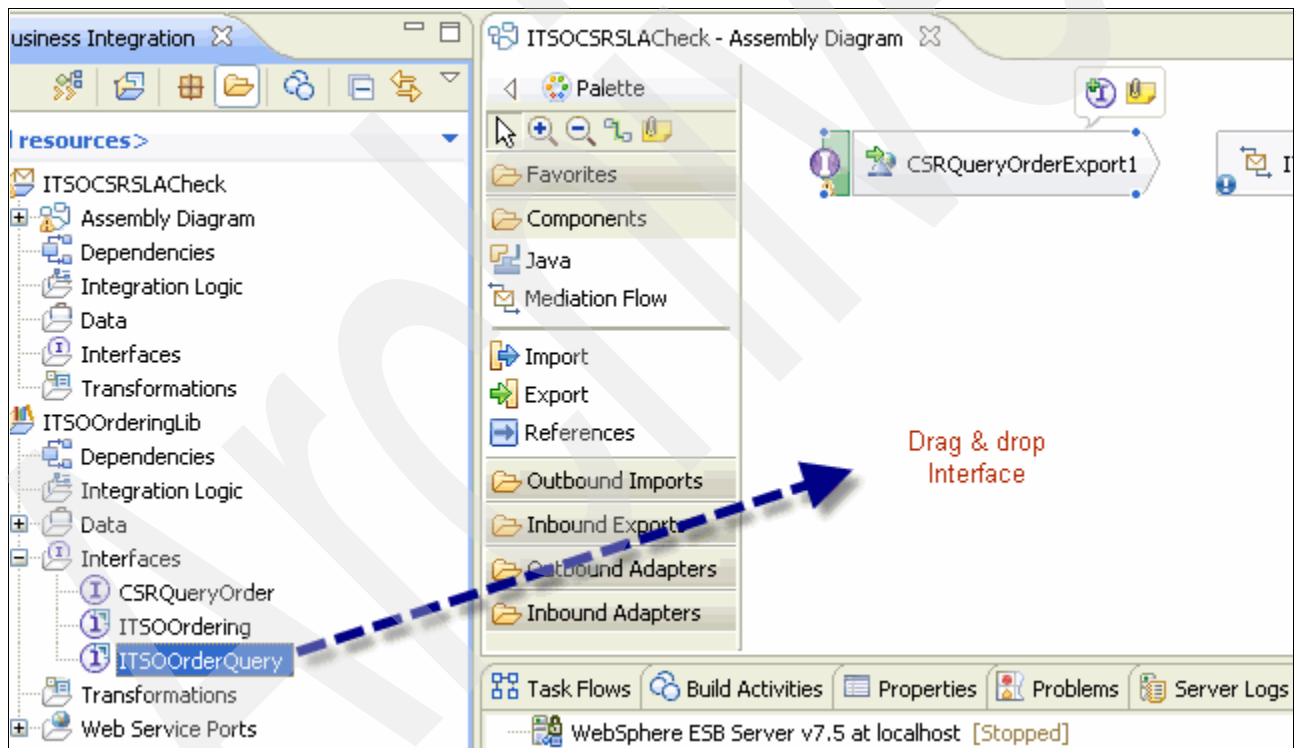


Figure 4-51 Drag reference interface to the Assembly Diagram

6. In the Component Creation dialog box, as shown in Figure 4-52 on page 186, select **Import with Web Service Binding** and click **OK**.

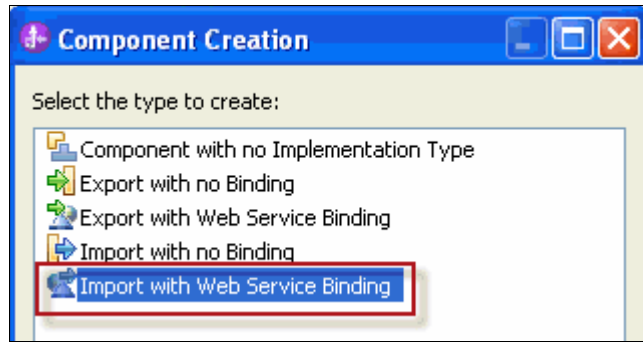


Figure 4-52 Component Creation

- In the ITSOrderQueryImport1 Web Service Import Details window, as shown in Figure 4-53, browse for the **ITSOrderQuerySOAP** interface from the ITSOrderingLib and click **OK**.

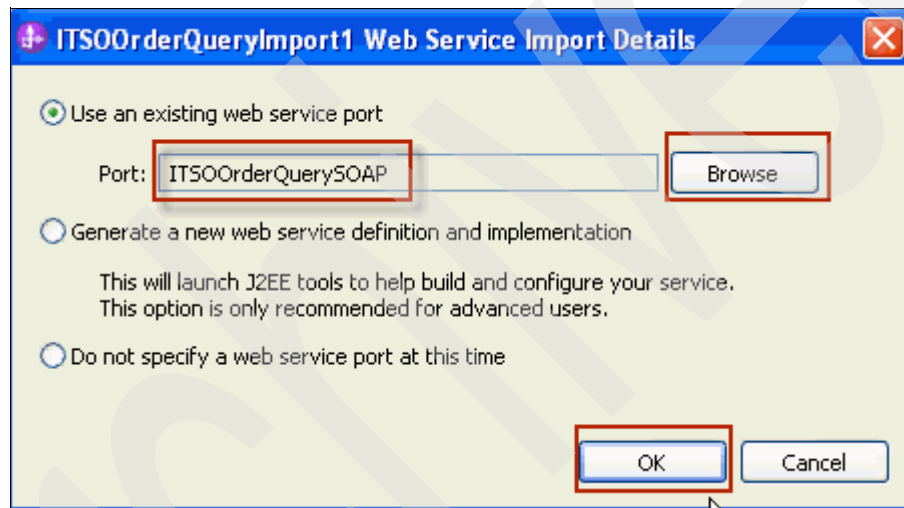


Figure 4-53 Select the Web service port

- In the next window, click **Finish**. You see the reference interface in the Assembly Diagram, as shown in Figure 4-54.

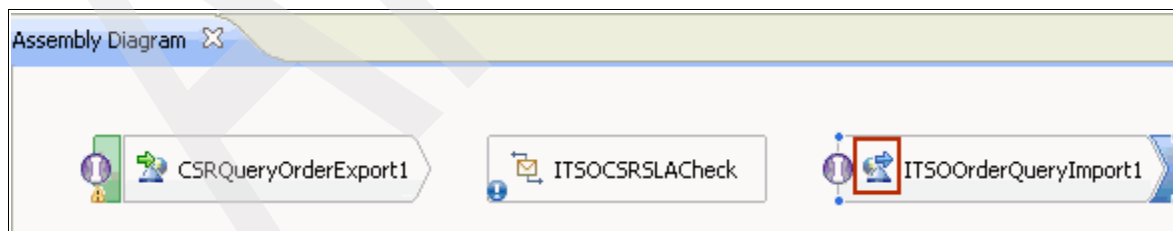


Figure 4-54 Assembly Diagram

- After wiring the interface and the reference with the mediation flow, the Assembly Diagram looks like Figure 4-55 on page 187.

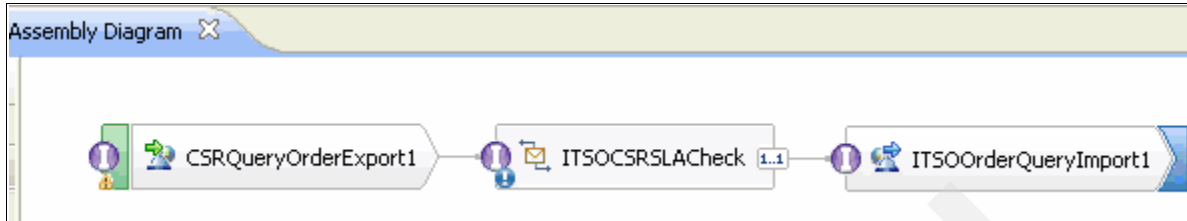


Figure 4-55 Assembly Diagram with wired components

Developing the mediation flow

We develop the mediation flow, which contains the main logic for the endpoint lookup into WebSphere Service Registry and Repository, and route the request based on the result from WebSphere Service Registry and Repository.

Follow these steps to implement the mediation module:

1. In the Assembly Diagram, double-click **ITSOCRSRLACheck**. Click **Yes** on the window that opens.
2. On the ITSOCRSRLACheck Overview tab, you see the interface and the references with the operations, as shown in Figure 4-56.

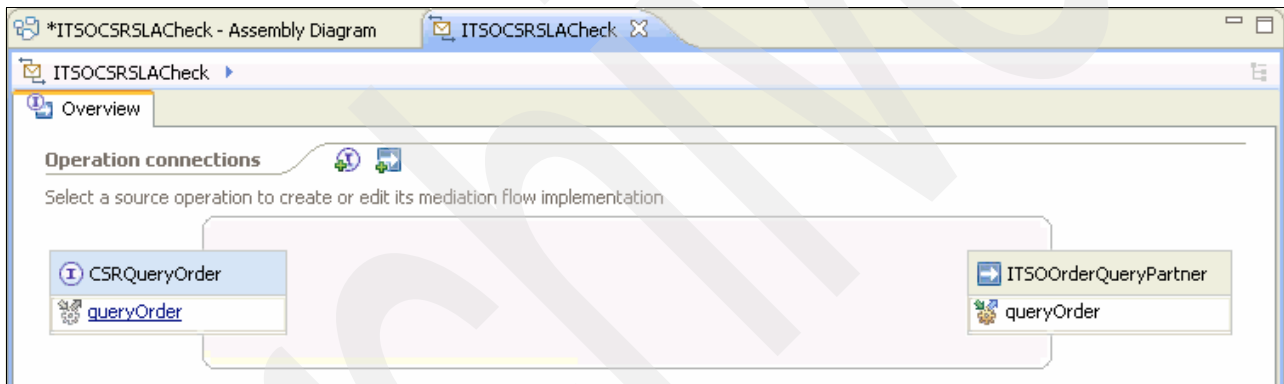


Figure 4-56 Operation connections window

3. Right-click **queryOrder** and select **Open Implementation**, as shown in Figure 4-57.

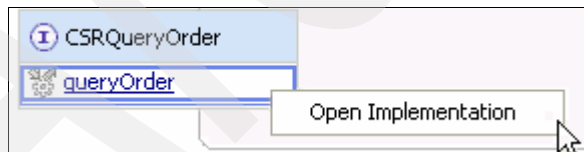


Figure 4-57 Query order

4. Select **Service Integration**, as described in Figure 4-58 on page 188.

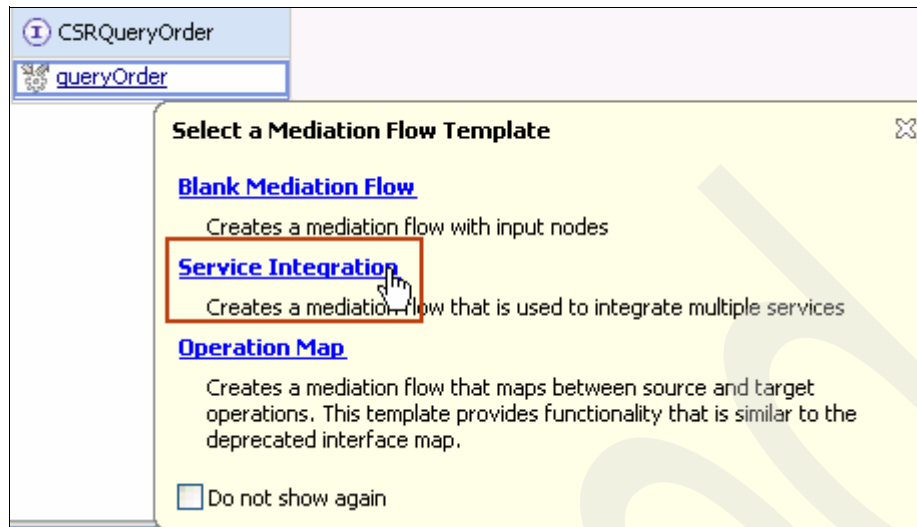


Figure 4-58 Service integration

5. In the Select the Services window, add the Reference and Target Operation and click **OK**, as shown in Figure 4-59.

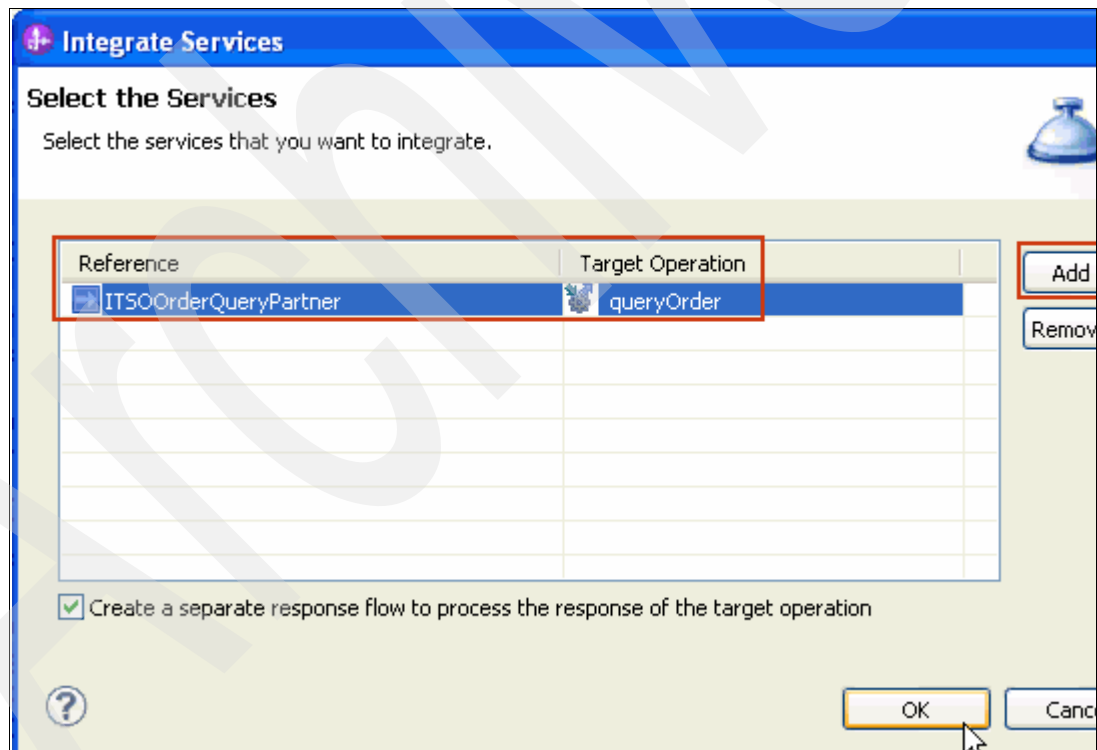


Figure 4-59 Select the services

Request flow

After selecting the services, you see the request flow where you can implement the request mediation logic.

Follow these steps to create the request flow in the mediation:

1. In the Request window, as described in Figure 4-60, add the mediation primitives, such as the Message Logger, SLAEndpointLookup, XSL Transformation map, and Fail primitives.

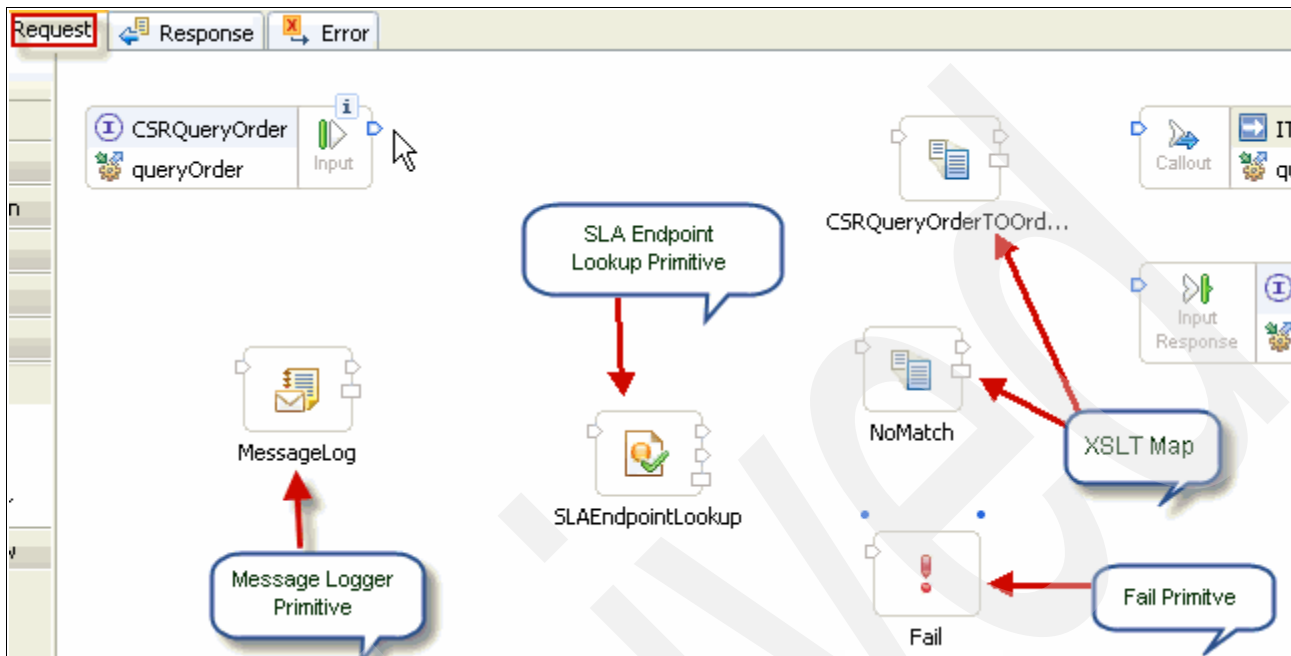


Figure 4-60 Request mediation flow

2. Wire each mediation primitive, as described in Figure 4-61.

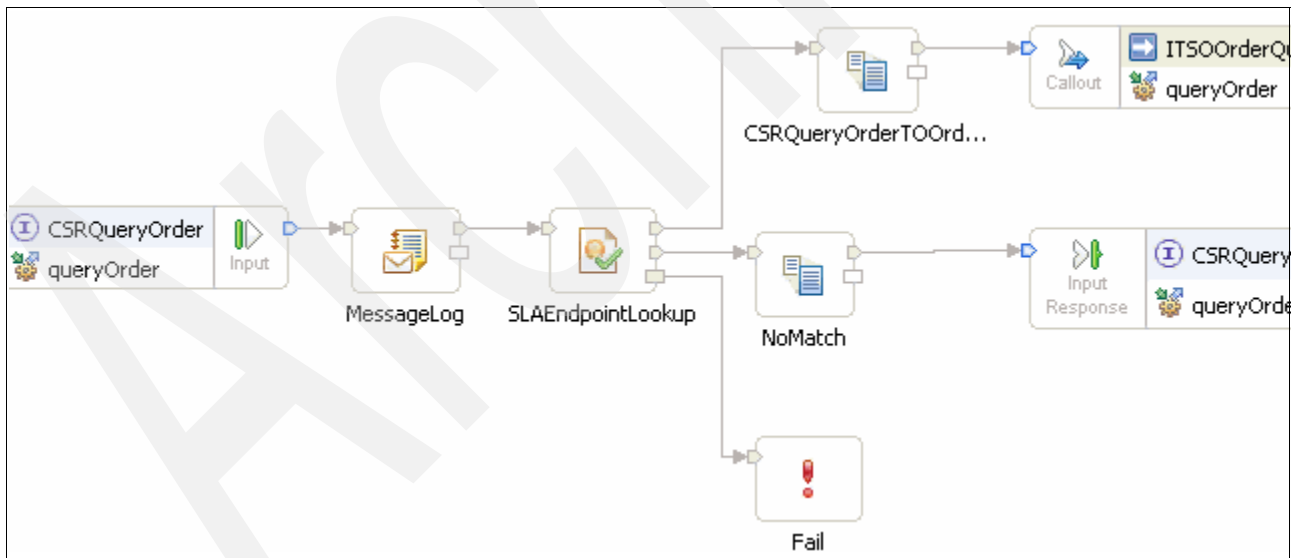


Figure 4-61 Wired mediation primitives in request flow

3. Click the **MessageLog** primitive, go to **Properties** → **Details**, and set the values, as described in Figure 4-62 on page 190.

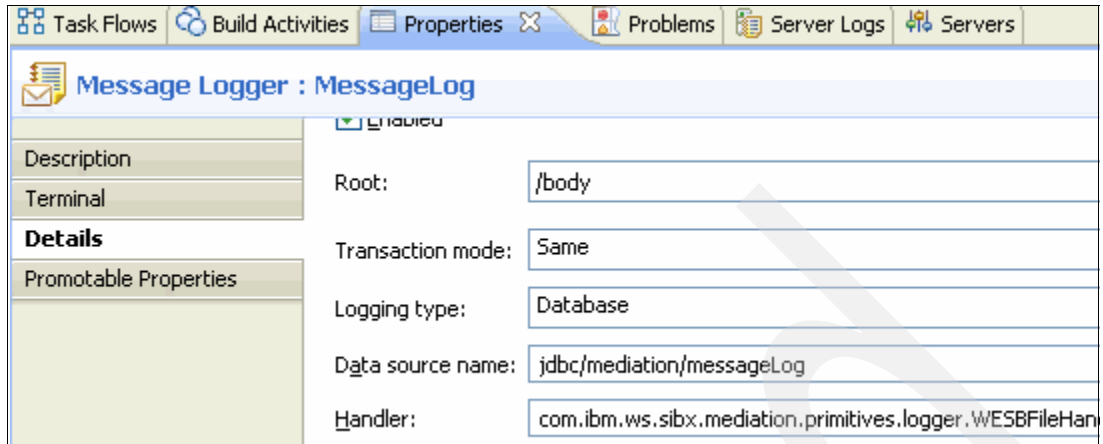


Figure 4-62 MessageLog properties

4. Click the **SLAEndpointLookup** primitive, go to **Properties** → **Details**, and set the values, as described in Figure 4-63.

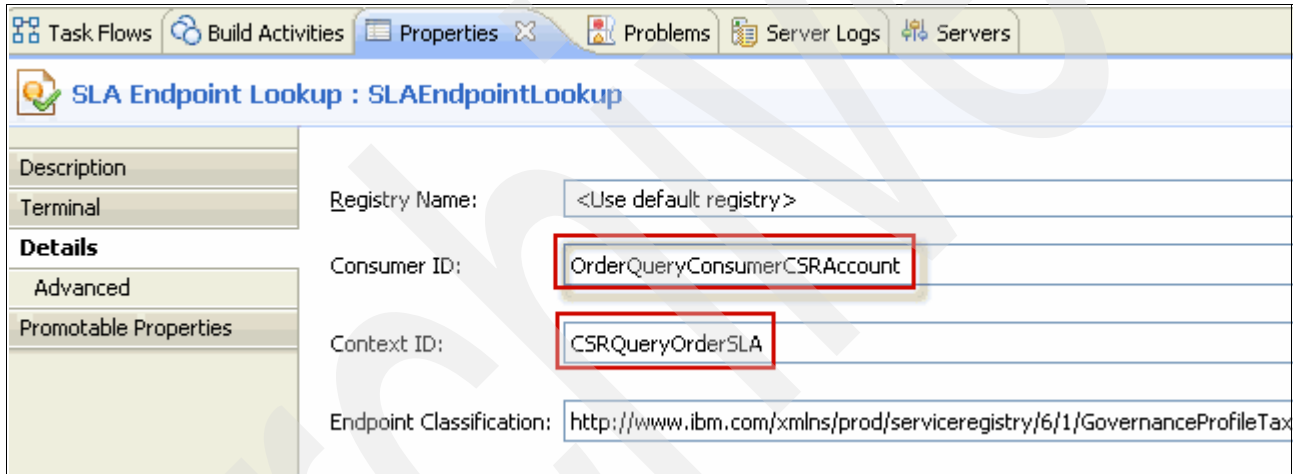


Figure 4-63 SLAEndpointLookup properties

5. Double-click the **CSRQueryOrderTOOrderQuery** XSLT map primitive, enter the name as shown in Figure 4-64, and click **Next**.

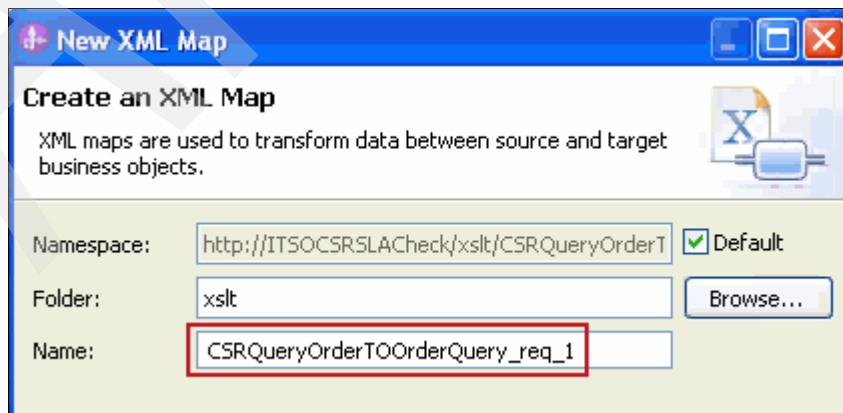


Figure 4-64 Create an XML Map

- In the Specify Message Types window, as described in Figure 4-65, select the Message Root as **/body** and click **Finish**.

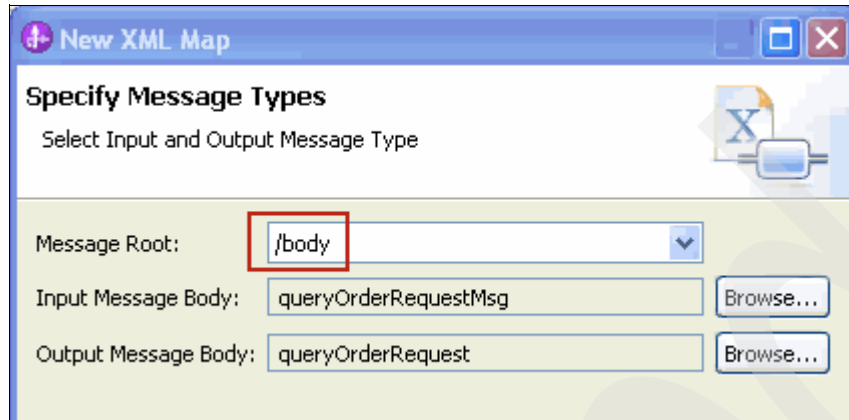


Figure 4-65 Specify Message Types

- Map the source **CustomerID** to the target **CustomerID** using the **Move** transform, as shown in Figure 4-66.

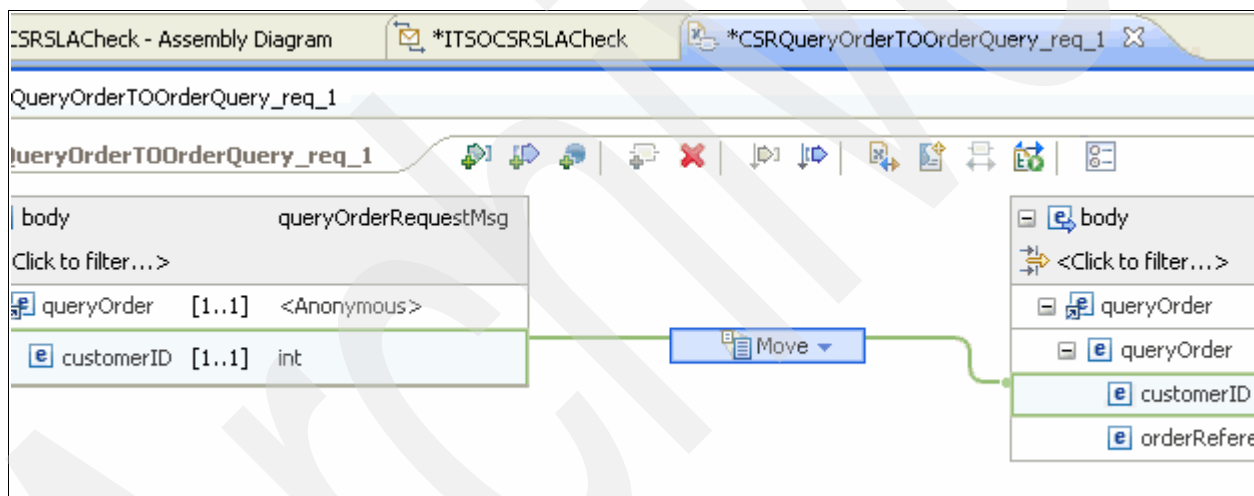


Figure 4-66 XSLT mapping for CSRQueryOrderTOOrderQuery

- Go back to the mediation flow and double-click the **NoMatch** XSLT map. Enter the name of the map and click **Next**. Then, for the Message Root, select **/body**, as described in Figure 4-67 on page 192.

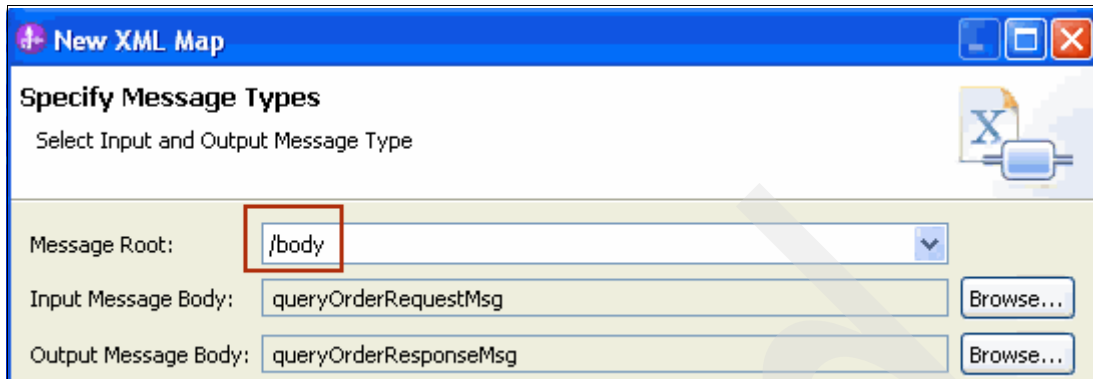


Figure 4-67 New XML Map

- As described in Figure 4-68, map the source **CustomerID** to the target **CustomerID** using the **Move** transform and to use for testing. Assign a value to the **customerType** field in case there is no match found from the SLAEndpointLookup primitive.

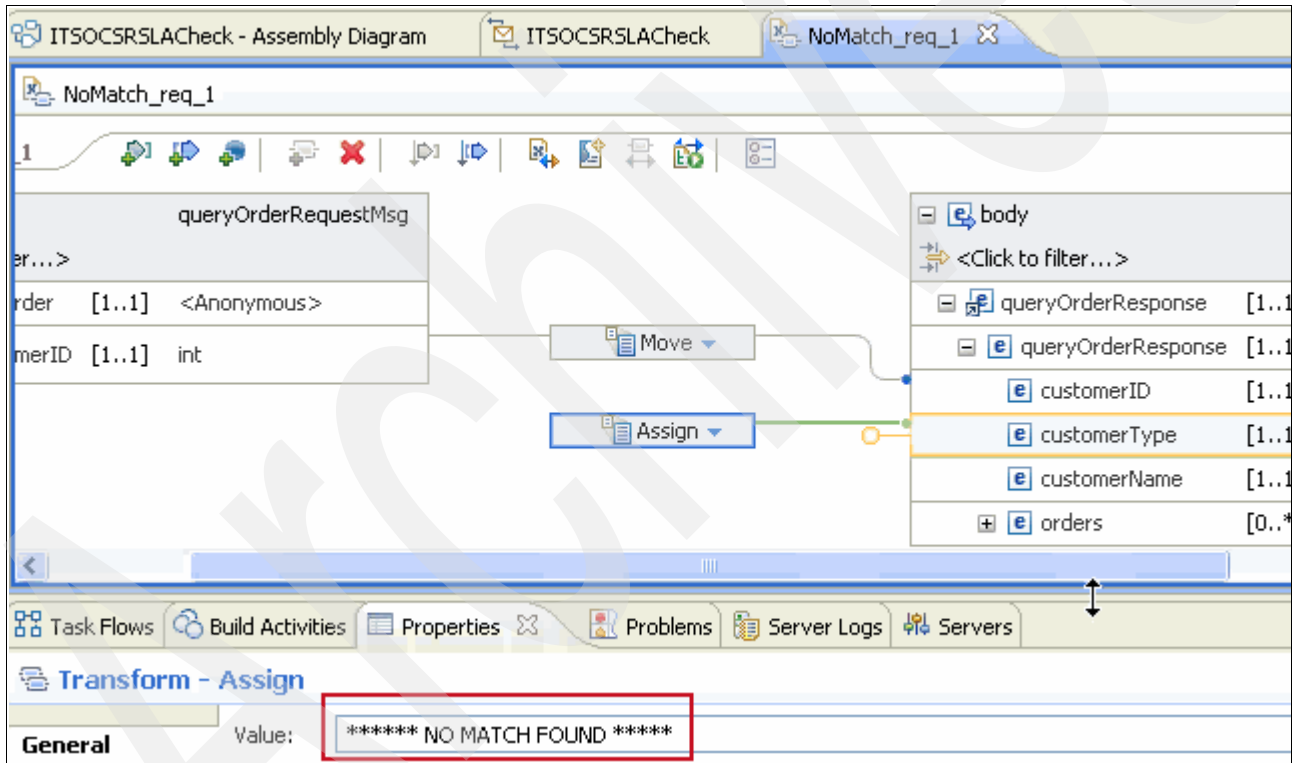


Figure 4-68 XSLT map for NoMatch

- Finally, the request flow looks like Figure 4-69 on page 193. Click **Response**.

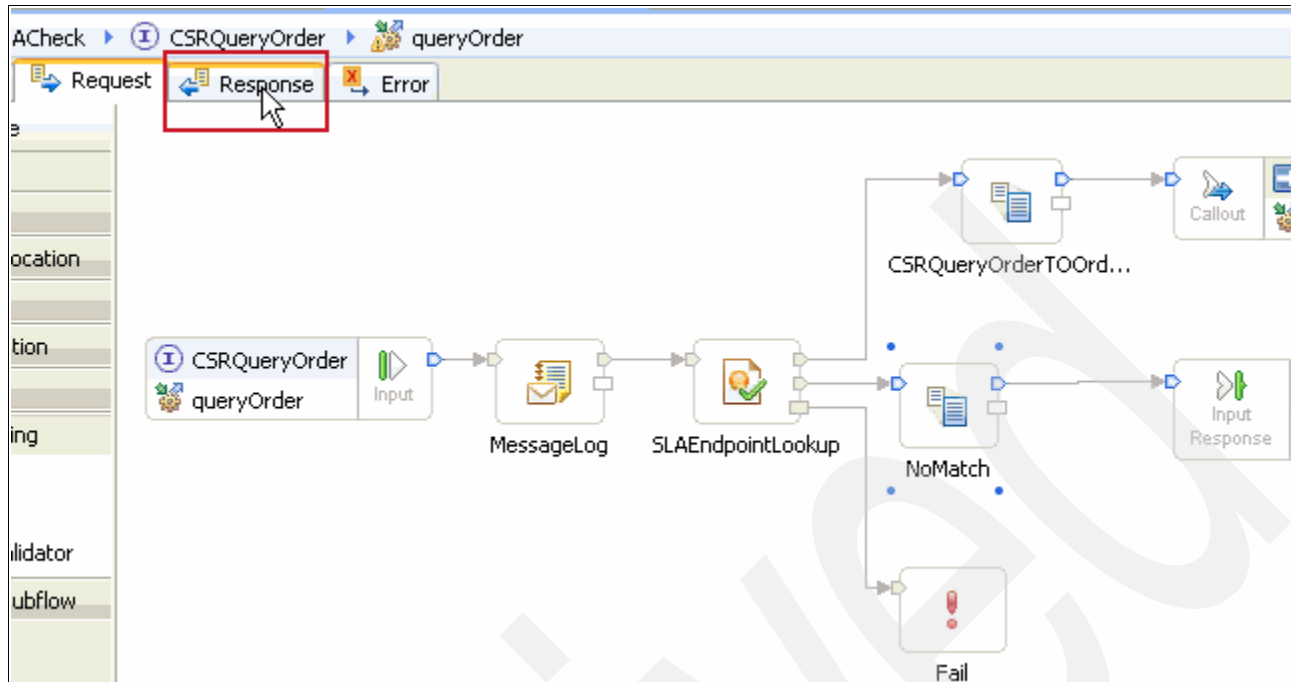


Figure 4-69 Request flow

Response flow

In the response flow, we implement the logic to transform the response message, which comes from the reference partner.

Follow these steps to develop the response flow:

1. In the response flow, drag the Trace Primitive and the XSL Transformation primitive to the Assembly Diagram. Then, rename and wire each primitive, as described in Figure 4-70.

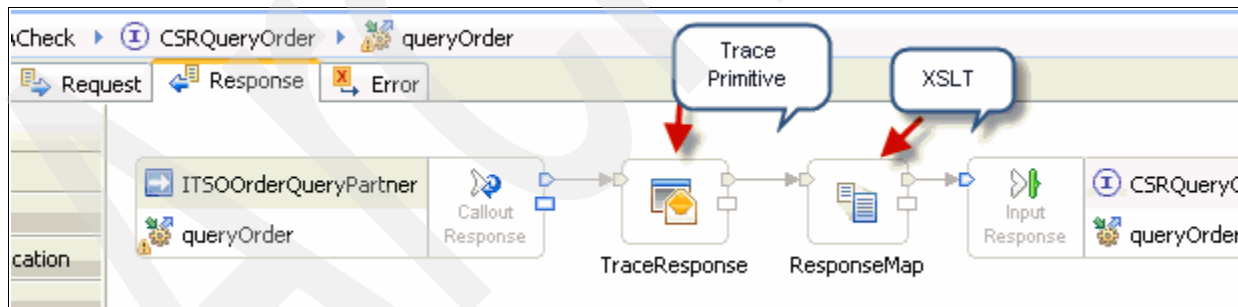


Figure 4-70 Response flow

2. Click the **TraceResponse** primitive, go to **Properties** → **Details**, and set the values, as described in Figure 4-71 on page 194.

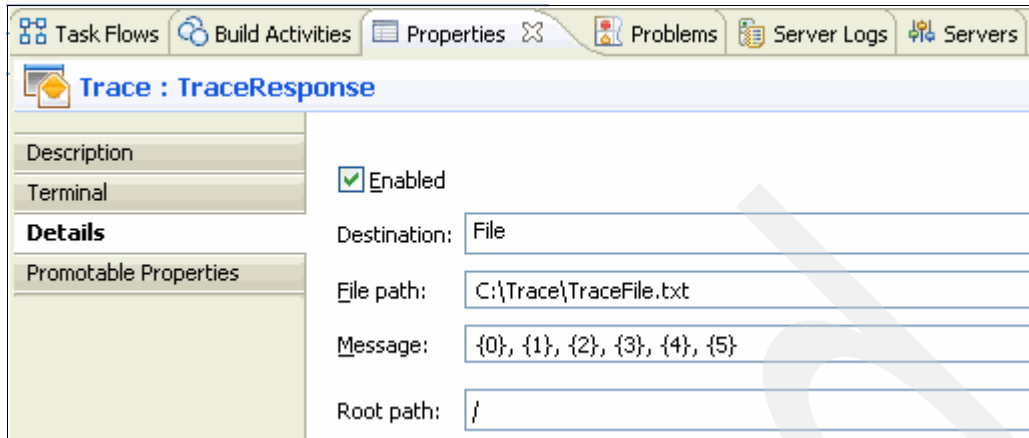


Figure 4-71 Trace primitive properties

3. Double-click the **ResponseMap** primitive and enter the name, as described in Figure 4-72.

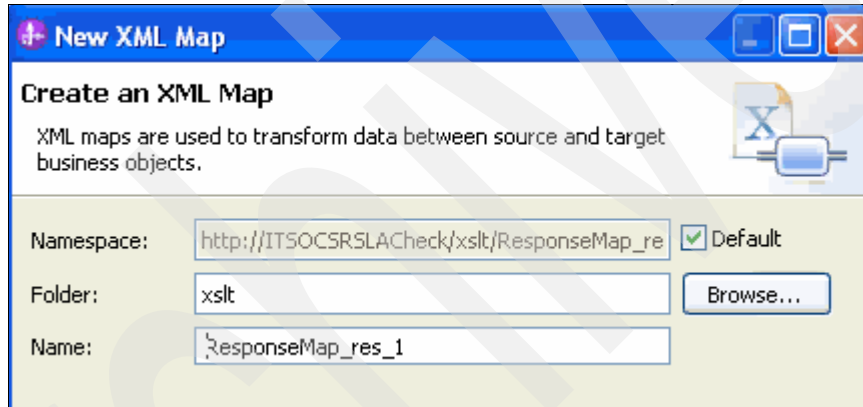


Figure 4-72 Create an XML Map

4. In the Specify Message Types window, as described in Figure 4-73, for the Message Root, select **/body**. Click **Finish**.

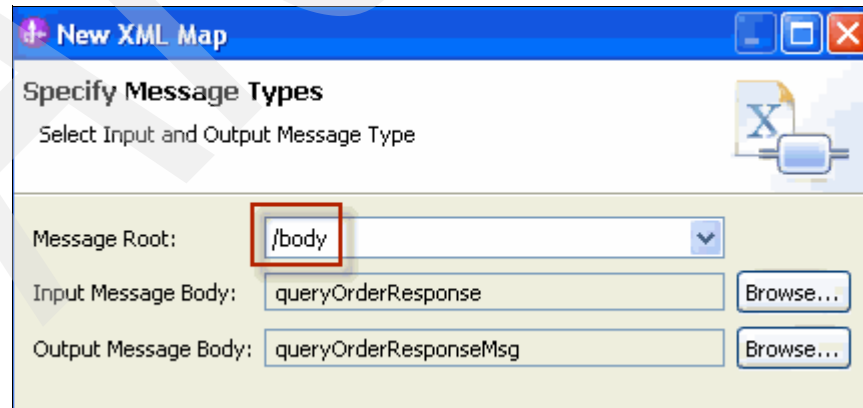


Figure 4-73 Specify Message Types

- In the mapping editor, map **queryOrderResponse** from the source to the target, as described in Figure 4-74.

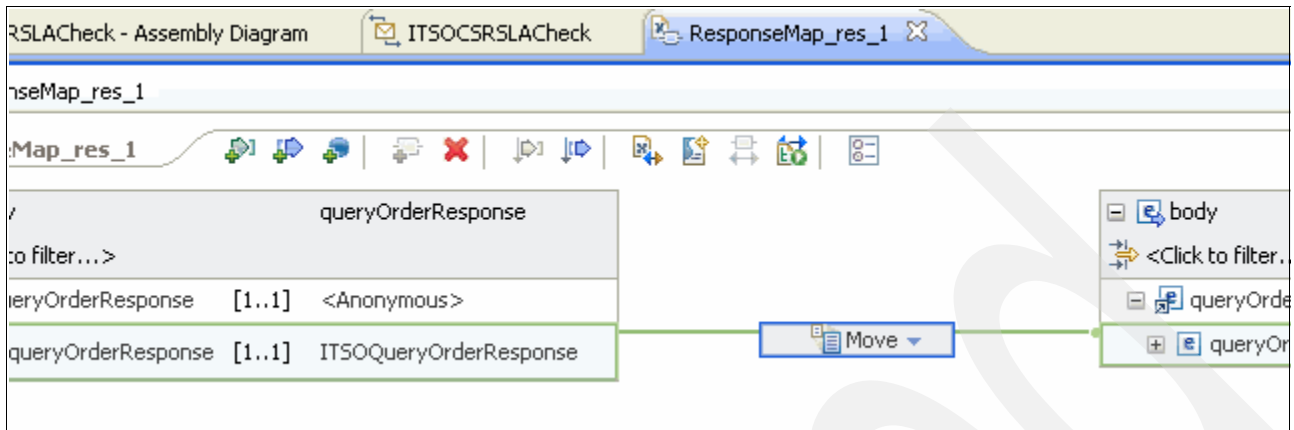


Figure 4-74 XSLT map for ResponseMap

- To capture the failure message in case the reference partner is not available or another error occurs, place the fail primitive named **Fail_Response** on the Assembly Diagram. The response flow looks like Figure 4-75.

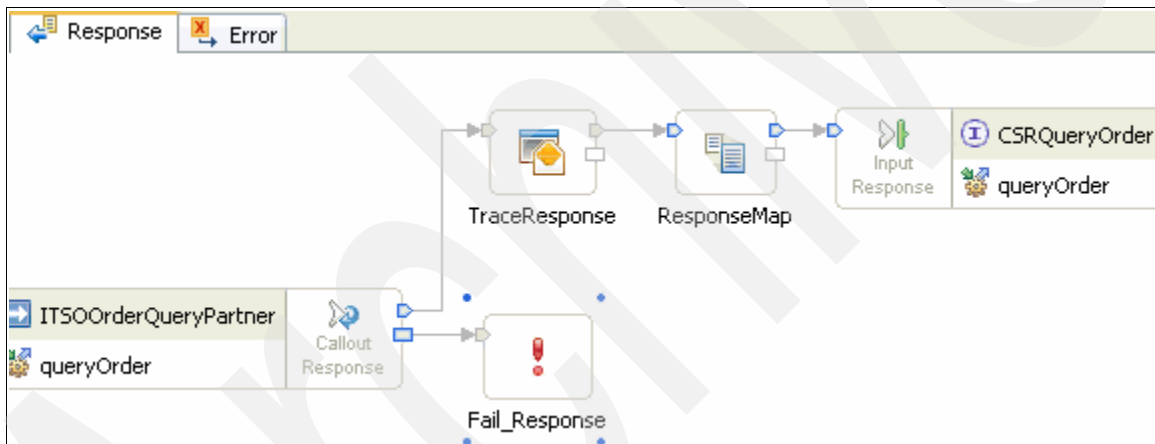


Figure 4-75 Response flow

Testing the mediation module

In this section, we demonstrate testing the mediation module that we developed for the CSR to access the ITSO ordering system.

Deploying the mediation module

Follow these steps to deploy the mediation module on the ESB Server:

- Start **WebSphere ESB Server v7.5** by clicking **Start**, as shown in Figure 4-76.



Figure 4-76 Start WebSphere ESB Server v7.5

2. Right-click **WebSphere ESB Server v7.5** and click **Add and Remove**, as described in Figure 4-77.

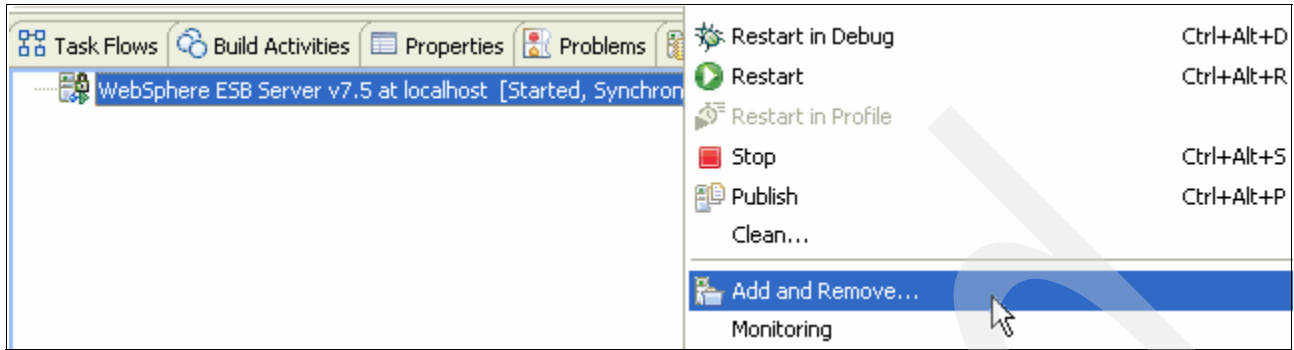


Figure 4-77 Add and Remove mediation module

3. In the Add and Remove window, select **ITSOCSRSLAcheckApp**, and click **Add >**, and click **Finish**, as described in Figure 4-78.

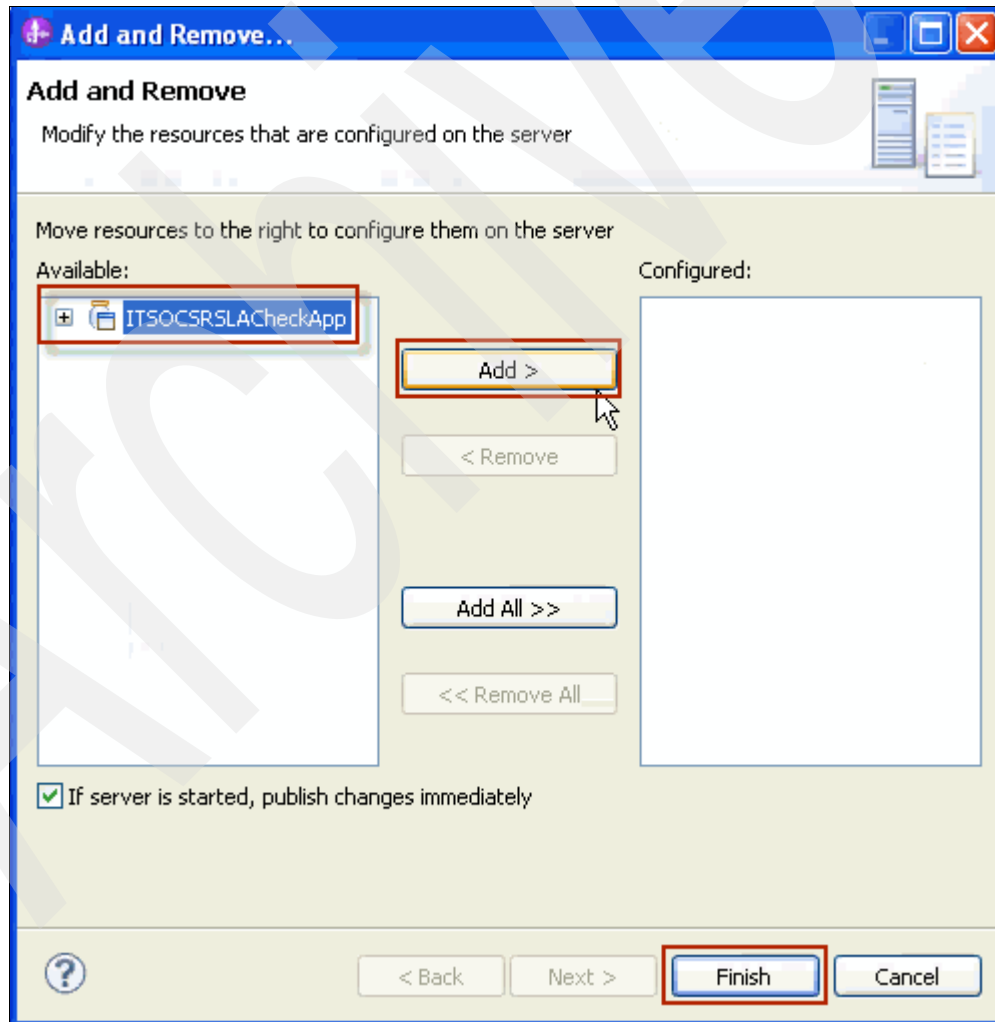


Figure 4-78 Add and Remove

- After successfully deploying ITSOCRSRLACheckApp on the WebSphere ESB Server, the window looks like Figure 4-79.

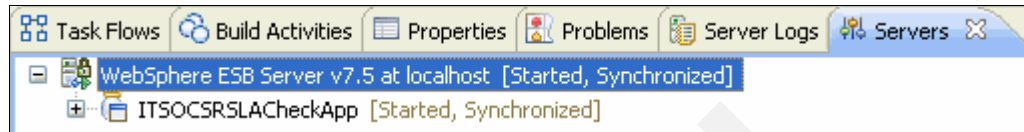


Figure 4-79 After deployment on ESB Server

- As described in Figure 4-80, you also get the following success message in the console, “Application started: ITSOCRSRLACheckApp”.

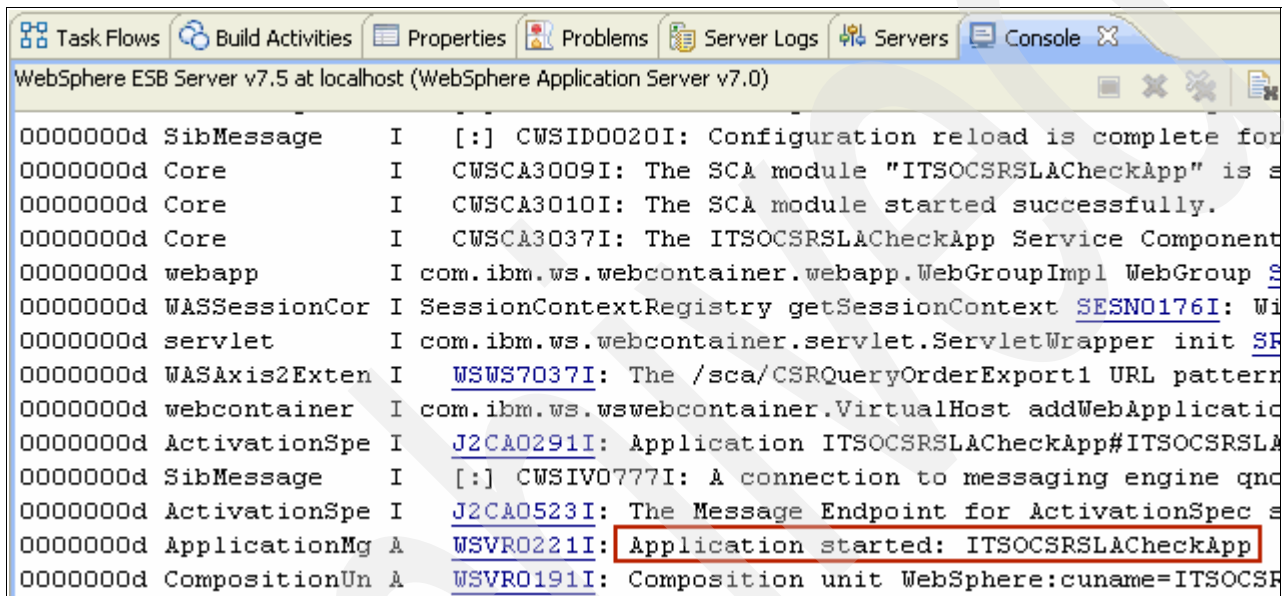


Figure 4-80 Console

Testing the request message

You can use Web service testing utilities to point to the ITSOCRSRLACheck web service. Example 4-1 shows the SOAP request message, which contains a customerID that the CSR can use to query the order.

Example 4-1 SOAP request message to query an order for a specific customerID

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:csr="http://ITS0Enterprise.com/CSRQueryOrder">
  <soapenv:Header/>
  <soapenv:Body>
    <csr:queryOrder>
      <customerID>10107</customerID>
    </csr:queryOrder>
  </soapenv:Body>
</soapenv:Envelope>
```

Response message

As a result, we get a response message with all the order details for the specific customerID, as shown in Example 4-2 on page 198.

Example 4-2 SOAP response message: Order details for a specific customerID

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <out:queryOrderResponse xmlns:x5xs="http://www.w3.org/2001/XMLSchema"
xmlns:io9="http://www.w3.org/2005/08/addressing"
xmlns:io8="http://www.ibm.com/xmlns/prod/websphere/http/sca/6.1.0"
xmlns:io7="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:io6="http://www.ibm.com/xmlns/prod/websphere/mq/sca/6.0.0"
xmlns:io5="http://www.ibm.com/websphere/sibx/smo/v6.0.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:out="http://ITSOEnterprise.com/CSRQueryOrder"
xmlns:io4="http://www.ITSOEnterprise.com/data/queryOrderResponse/Orders"
xmlns:io3="http://www.w3.org/2003/05/soap-envelope"
xmlns:io2="http://www.ITSOEnterprise.com/data/lineitem"
xmlns:io="http://www.ITSOEnterprise.com/data/queryOrderResponse"
xmlns:out2="wsdl:http://ITSOEnterprise.com/CSRQueryOrder">
      <queryOrderResponse
xmlns:tns="http://www.ITSOEnterprise.com/ITSOOrdering/">
        <customerID>10107</customerID>
        <customerType>gold</customerType>
        <customerName>CustomerCare</customerName>
        <orders orderReferenceNumber="1023" orderDate="DATE '2011-06-27'"
xmlns:qrons="http://www.ITSOEnterprise.com/data/queryOrderResponse/Orders">
          <orderDetails
xmlns:lns="http://www.ITSOEnterprise.com/data/lineitem">
            <productID>12345</productID>
            <productDesc>Webcam</productDesc>
            <numberItem>1</numberItem>
            <unitPrice>2.3445E+2</unitPrice>
          </orderDetails>
          <orderDetails
xmlns:lns="http://www.ITSOEnterprise.com/data/lineitem">
            <productID>12346</productID>
            <productDesc>keyboard</productDesc>
            <numberItem>1</numberItem>
            <unitPrice>9.999E+1</unitPrice>
          </orderDetails>
          <totalPayment>3.3444E+2</totalPayment>
        </orders>
      </queryOrderResponse>
    </out:queryOrderResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

4.4 Summary

WebSphere Service Registry and Repository plays a key role in the end-to-end governance underpinnings of the SOA life cycle. WebSphere Service Registry and Repository establishes a central point for finding and managing service metadata that has been acquired from several sources, including service application deployments, other service metadata, endpoint registries, and repositories, such as Universal Description, Discovery, and Integration (UDDI). In WebSphere Service Registry and Repository, the registry stores information about life

cycle status and other metadata that is used to filter the visibility of endpoints according to usage and context. This metadata includes information about the development, assembly, testing, deployment, operation, and version number.

In this chapter, we discuss how ITSO Enterprise reuses the ordering system with the help of IBM WebSphere ESB and how to build a service gateway using IBM WebSphere Service Registry and Repository. ITSO Enterprise added an SLA in WebSphere Service Registry and Repository for the CSR service to view the orders that are associated with each customer within the CSR application.

In this chapter, we described how the WebSphere Service Registry and Repository implementation can address the goals of improved business agility, reach the SLA, optimize cost, take advantage of new business environments, and respond to new opportunities quickly and effectively.

Archived



Message Privacy Enforcement pattern

Information technology security is a vital component of business success. Moreover, current technologies and the expansion of global markets for small devices, wireless networks, shared infrastructure, cloud computing, and other upcoming changes in IT are forcing businesses and governments to create processes and tighten the requirements for security. Any company that wants to demonstrate trust to its customers and partners must take security seriously.

Adherence to security policy requirements, such as the Sarbanes-Oxley Act (SOX), Health Insurance Portability and Accountability Act (HIPAA), and the Payment Card Industry Data Security Standard (PCI DSS), is still an unknown area for multiple companies. If you research these requirements, you notice that most of what is requested is not something on which a product can be certified. Security compliance activities are related to several human processes that are there to ensure end-to-end security for applications and information across multiple hardware and software platforms and networks.

By abiding to these processes, the intention is to decrease or put an end to security breaches that can lead to the loss or misuse of private data, such as medical records and credit card information, and fraud, losses in revenue, and other business impacts for both customers and the company.

In this chapter, we introduce you to a smart solution for a PCI DSS specific requirement that ITSO Enterprise has to implement.

This chapter contains the following sections:

- ▶ 5.1, “Introduction to the business scenario” on page 203
- ▶ 5.2, “Architecture diagram” on page 204
- ▶ 5.3, “Benefits” on page 207
- ▶ 5.4, “Technical implementation” on page 208
- ▶ 5.5, “Summary” on page 232

How to recreate these scenarios: You can download the configuration files and programs that are used in this chapter from the ITSO Redbooks FTP site. You can use these files to recreate these scenarios in your environment. For download instructions, refer to Appendix A, “Additional material” on page 401.

Archived

5.1 Introduction to the business scenario

The ITSO Enterprise Point-of-Sale (POS) and Back-end authorization service works as a hub and spoke, currently exchanging messages through a messaging system. Every store has an infrastructure to support sending messages to the back-end systems. In this case, the PoS service puts an order message in a queue with the customer's credit card information. This message is taken from the queue by the Enterprise Service Bus (ESB) that will further process the order and call the authorization service that has been provided by a third-party company. After the order is authorized and processed, the ESB puts the reply message on the store queue, but not before the authorization and processing, to make sure that the ESB defines a correlation with the request message.

A secure connection is in place to exchange messages between the stores and the back-end system. An authentication/authorization mechanism has been configured to ensure that only authorized users can access the queues. See Figure 5-1.

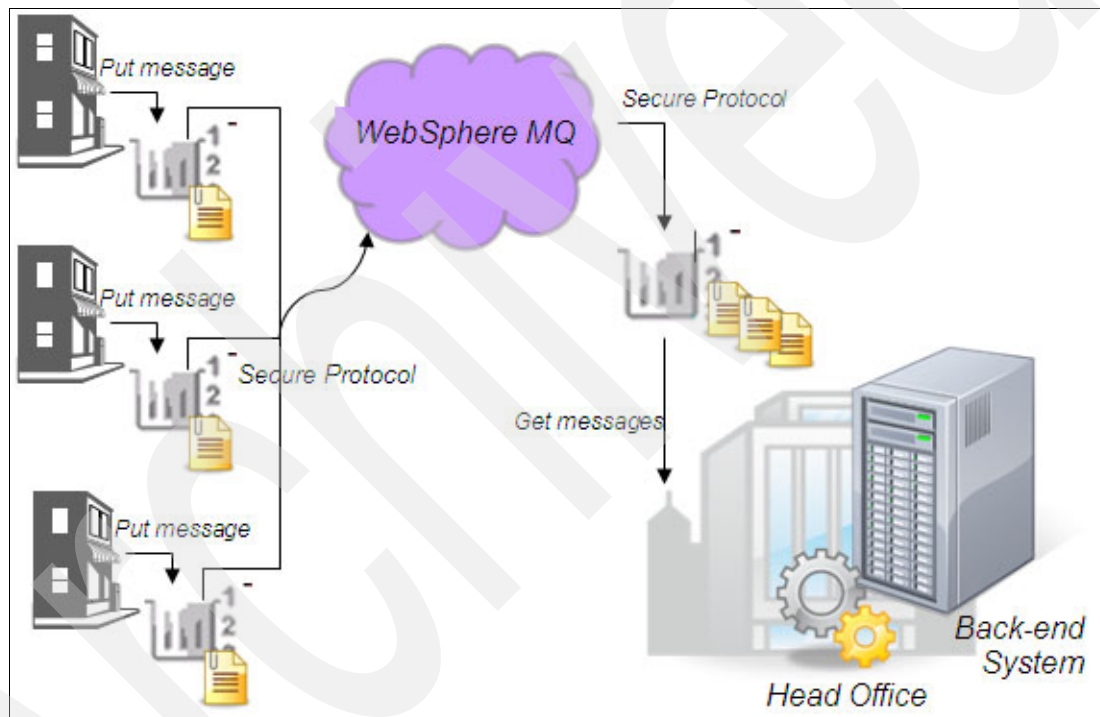


Figure 5-1 ITSO Enterprise hub and spoke design: Stores and back-end system

5.1.1 Business requirements

PCI DSS is a mandatory requirement for ITSO Enterprise. ITSO Enterprise must create or change its processes and applications to comply with the security policy standards and technical specifications.

One of the problems that has been identified by the IT Architect in charge of the project is that messages are being stored in the queues without encryption and that must change. The encryption of the credit card number is mandatory when the credit card number is stored.

The project has to be implemented within a fixed period of time. Otherwise, ITSO Enterprise can be charged with additional fees for all credit card transactions and, in certain cases, penalties that are based on specific criteria agreed to between ITSO Enterprise and the authorization services.

In addition to the message privacy requirement, ITSO Enterprise must keep audit logs and traces to satisfy other PCI DSS standards.

5.1.2 Available options

In the past, ITSO Enterprise developers rewrote the applications that were involved to comply with the requirements. This approach led the company to undesirable results, such as developers who knew the applications leaving the company, which left the remaining developers to maintain that code. Several times, bugs were identified in these in-house applications, which usually took more time to resolve than expected by ITSO Enterprise. Although all these issues have happened in the past, in-house development is still an option.

The other option is to replace the messaging system with other infrastructure, which will require significant effort. Sizing the project, including the application changes in the code, the testing phase, and the deployment phase, determined that both the target date and budget were at risk.

The IT Architect discovered Smart SOA when researching existing solutions for this problem. The IT Architect identified a product to fit exactly what ITSO Enterprise needed to protect messages inside queues without changing application code.

5.2 Architecture diagram

The design of the solution will keep the existing infrastructure and add additional layers between clients and queues. These layers are interceptors, which will capture messages before they are stored in the queue, to encrypt them and before they leave the queue to decrypt them. See Figure 5-2.

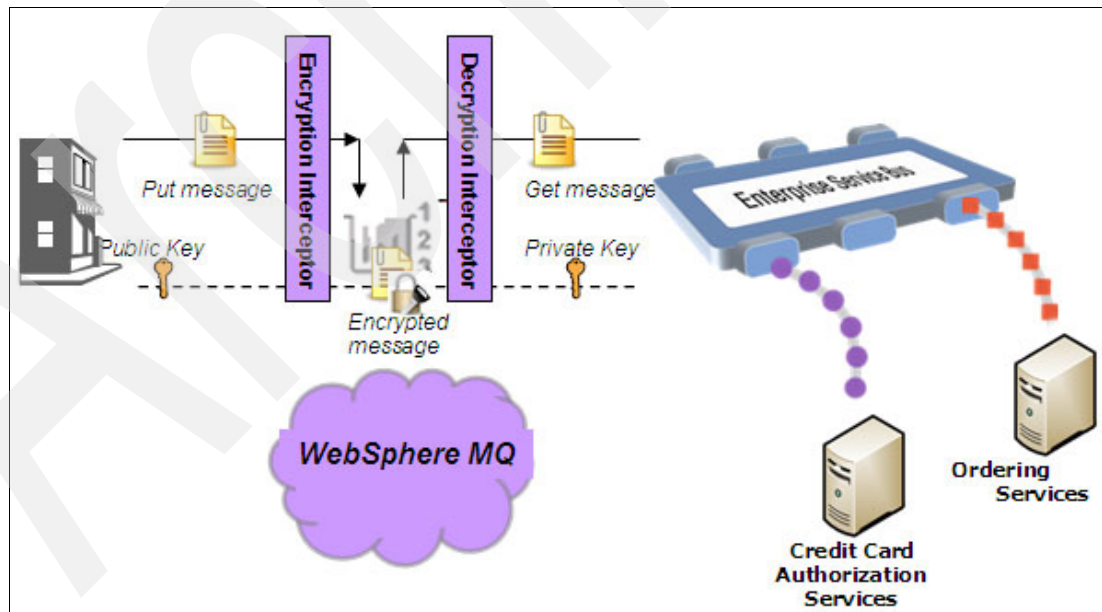


Figure 5-2 Architectural design

5.2.1 Pattern

The architecture for this solution contains several connectivity pattern implementations, but we focus on two: Message Privacy Enforcement and the service gateway.

Message Privacy Enforcement pattern

Message Privacy Enforcement includes interceptors to any messaging calls, such as open, close, get, and put. When one of these calls is intercepted, a policy is read to encrypt and decrypt the message as defined. Each operation has its own responsibilities, such as a get call, for instance, performs signature checking and decryption before it returns the original message to the calling client (Figure 5-3).

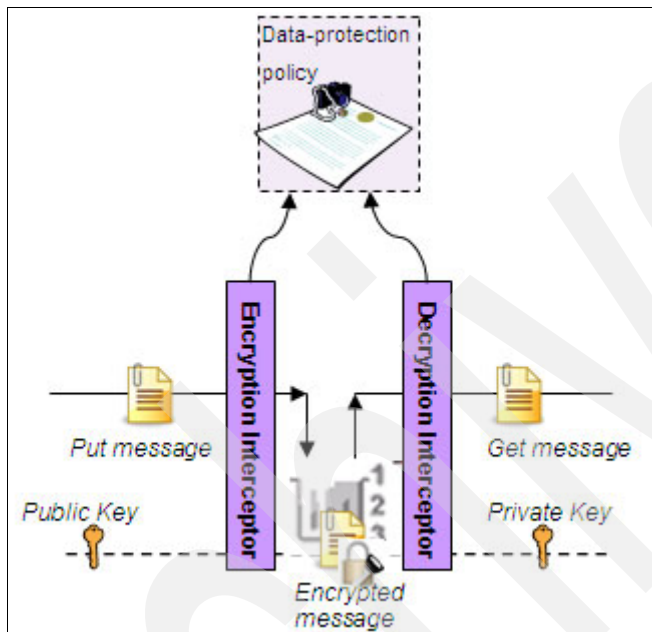


Figure 5-3 Message Privacy Enforcement

Service gateway

The service gateway is a known connectivity pattern. For this particular case, it works as a security gateway. It authenticates, authorizes, and fulfills the requested audit requirements (Figure 5-4 on page 206).

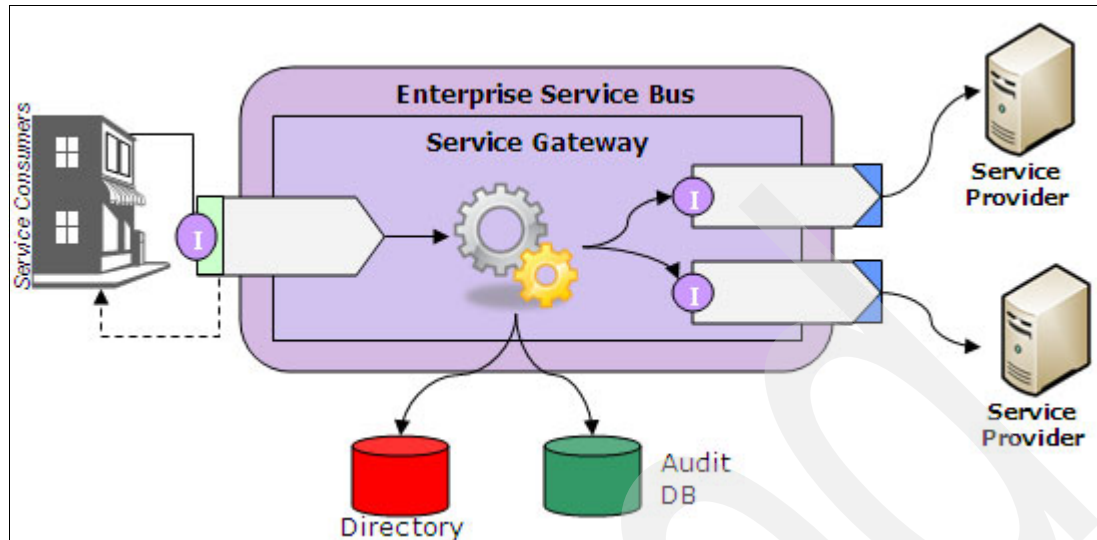


Figure 5-4 Security gateway

5.2.2 Products used

ITSO Enterprise decided to go with the Smart Service-Oriented Architecture (SOA) solution. The only product that the company must include in its existing infrastructure is WebSphere MQ Advanced Message Security. See “IBM WebSphere MQ Advanced Message Security” on page 6 for more information.

5.2.3 Key features

WebSphere MQ Advanced Message Security provides the following key features:

- ▶ Protected message content end-to-end by encrypting data in queues
- ▶ Reduced operational costs with centralized management of access to WebSphere MQ queues, auditing, and the ability to consolidate secure transfer and message transfer into one single transport layer
- ▶ Increased flexibility so changes to applications, hardware, or networks do not require reworking security code
- ▶ Improved operational savings and simplification
- ▶ Reduced administrative effort
- ▶ Reduced skill requirements and maintenance
- ▶ Improved reliability and auditability

5.2.4 Solution diagram

ITSO Enterprise implemented the Smart SOA solution. Figure 5-5 on page 207 shows the ITSO Enterprise stores and ordering systems integration.

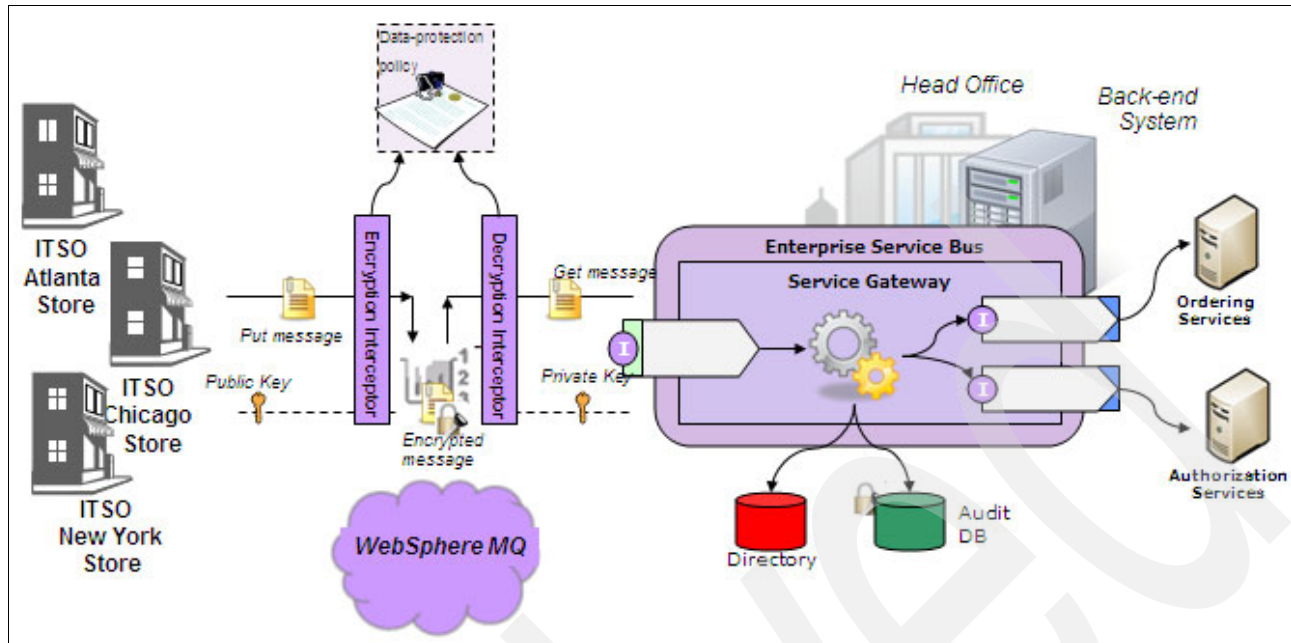


Figure 5-5 Solution diagram

5.3 Benefits

The ITSO Enterprise solution provides the following benefits:

- ▶ Reduced time and cost needed to comply with the security requirement by applying WebSphere MQ Advanced Message Security in its existing MQ network.
- ▶ Centralized management of security policies across the WebSphere MQ assets.
- ▶ Further extended policies to meet other security regulatory requests or even improve security on other communication channels that use or will use WebSphere MQ, such as file transfer solutions. See Chapter 6, "Enterprise Messaging and File Backbone pattern" on page 233.
- ▶ Avoided homemade security mechanisms that are built on top of WebSphere MQ, which require specialist skills and can be complicated and expensive to maintain.
- ▶ Immediate return on investment (ROI) upon deployment by securing the existing applications without application changes.
- ▶ Increased flexibility of WebSphere MQ network so that changes to hardware, OS, applications, and networks do not require changes to the existing WebSphere MQ network or applications.
- ▶ Ability to meet regulatory targets, avoiding penalties or losing the ability to process credit card transactions.
- ▶ Improved effect on company image by closing security breaches.

5.4 Technical implementation

This section provides instructions about how to update a mediation flow, which is built on top of WebSphere Enterprise Service Bus (ESB) 7.5, that reads a WebSphere MQ queue, processes the message, and returns the response in a separate WebSphere MQ queue.

5.4.1 Assumptions

We assume that you have a basic knowledge of Java programming and Java Message Service (JMS) applications using WebSphere MQ. An understanding of WebSphere Integration Designer or WebSphere Integration Developer and WebSphere Application Server configuration are also beneficial.

This implementation is based on a single Microsoft Windows Server where we simulate both the client and server. We assume that you have administrative skills on Microsoft Windows that allow you to create users and groups and issue commands in a command line.

5.4.2 Prerequisites

Based on the assumptions, we require that you already have installed software. You need to know how to create simple artifacts in the software listed that you will use further to implement the business scenario.

Products

For this chapter, we use the following product versions:

- ▶ IBM WebSphere MQ V7.0.1.4 (Client and Server)
- ▶ IBM WebSphere MQ Advanced Message Security V7.0.1.1

Note: Be aware that certain features of WebSphere MQ Advanced Message Security (MQ AMS) have WebSphere MQ version dependency, such as the WebSphere MQ Advanced Message Security Explorer Plug-in. For all prerequisites, check this site:

http://publib.boulder.ibm.com/infocenter/mqams/v7r0m1/topic/com.ibm.mqese.doc/installing/installating_prerequisites.htm

- ▶ IBM Integration Designer V7.5 for WebSphere Enterprise Service Bus V7.5

IBM WebSphere MQ artifacts

During the implementation, we use one queue manager and two queues:

- ▶ Queue Manager: AMS.QM
- ▶ Queues: AMS.STORE.REQUEST.Q and AMS.STORE.REPLY.Q

IBM WebSphere ESB mediation flow

To make it easier to understand and test, the flow only reads the message from the request queue and replies to the same message in the reply queue. In the scenario, it calls other services, but they have been presented in the previous chapters. Refer to Chapter 2, “Service Enablement pattern” on page 23 and Chapter 3, “Service Enablement pattern: Enterprise Content Management System” on page 75 for more details. Also, auditing requirements vary among security policies.

Operating system users and groups

This chapter uses three users: the store sending the message, the WebSphere ESB mediation flow user, and the WebSphere MQ administrator:

- ▶ storeusr
- ▶ webusr
- ▶ Administrator

5.4.3 Setting up the environment

We configure IBM WebSphere Advanced Message Security to intercept the messages between the store and WebSphere ESB.

Configuring authorizations on MQ artifacts

We implement a Data Privacy policy. Only the sender (store) can put messages on the queue and only the receiver (back-end system) can read messages. To do so, we issue basic MQ authorization commands. Follow these steps:

1. Create a queue manager, as shown in Example 5-1.

Example 5-1 Create AMS.QM queue manager output

```
C:\Users\Administrator>cd\wmq\bin\  
C:\WMQ\bin>crtmqm AMS.QM  
WebSphere MQ queue manager created.  
Directory 'C:\WMQ\qmgrs\AMS!QM' created.  
Creating or replacing default objects for AMS.QM.  
Default objects statistics : 68 created. 0 replaced. 0 failed.  
Completing setup.  
Setup completed.
```

2. Start the queue manager, as shown in Example 5-2.

Example 5-2 Starting the queue manager output

```
C:\WMQ\bin>strmqm AMS.QM  
WebSphere MQ queue manager 'AMS.QM' starting.  
5 log records accessed on queue manager 'AMS.QM' during the log replay phase.  
Log replay for queue manager 'AMS.QM' complete.  
Transaction manager state recovered for queue manager 'AMS.QM'.  
WebSphere MQ queue manager 'AMS.QM' started.
```

3. Create a listener. See Example 5-3.

Example 5-3 Creating a listener output

```
C:\Users\Administrator>runmqsc AMS.QM  
5724-H72 (C) Copyright IBM Corp. 1994, 2009. ALL RIGHTS RESERVED.  
Starting MQSC for queue manager AMS.QM.  
  
DEFINE LISTENER(AMS.QM.LISTENER) TRPTYPE(TCP) PORT(1414)  
1 : DEFINE LISTENER(AMS.QM.LISTENER) TRPTYPE(TCP) PORT(1414)  
AMQ8626: WebSphere MQ listener created.  
start listener(AMS.QM.LISTENER)  
2 : start listener(AMS.QM.LISTENER)  
AMQ8021: Request to start WebSphere MQ Listener accepted.
```



```
end
    3 : end
2 MQSC commands read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

```
C:\Users\Administrator>
```

4. Create a channel. See Example 5-4.

Example 5-4 Creating a channel output

```
C:\Users\Administrator>runmqsc AMS.QM
5724-H72 (C) Copyright IBM Corp. 1994, 2009. ALL RIGHTS RESERVED.
Starting MQSC for queue manager AMS.QM.
```

```
DEFINE CHANNEL(AMS.CHANNEL) CHLTYPE(SVRCONN)
    1 : DEFINE CHANNEL(AMS.CHANNEL) CHLTYPE(SVRCONN)
AMQ8014: WebSphere MQ channel created.
end
    2 : end
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

```
C:\Users\Administrator>
```

5. Create the queues. See Example 5-5.

Example 5-5 Creating queues output

```
C:\WMQ\bin>runmqsc AMS.QM
5724-H72 (C) Copyright IBM Corp. 1994, 2009. ALL RIGHTS RESERVED.
Starting MQSC for queue manager AMS.QM.
```

```
define qlocal(AMS.STORE.REQUEST.Q)
    1 : define qlocal(AMS.STORE.REQUEST.Q)
AMQ8006: WebSphere MQ queue created.
end
    2 : end
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

6. Create the WebSphere MQ Advanced Message Security (MQ AMS) system queues. See Example 5-6.

Example 5-6 Creating MQ AMS system queues

```
C:\WMQ\bin>cd\WMQAMS\bin
C:\WMQAMS\bin>runmqsc AMS.QM < defineqs.mqs
5724-H72 (C) Copyright IBM Corp. 1994, 2009. ALL RIGHTS RESERVED.
Starting MQSC for queue manager AMS.QM.
: *****
: *
```



```

: * WebSphere MQ Advanced Message Security queues
: *
: * 5724-Z94
: * (c) Copyright IBM Corp. 2010
: * The source code for this program is not published or otherwise divested
: * of its trade secrets, irrespective of what has been deposited with the
: * U.S. Copyright Office.
: *
: *
:
*****
: *
1 : DEFINE QLOCAL(SYSTEM.PROTECTION.POLICY.QUEUE) MAXDEPTH(999999999)
MAXMSGL(4194304) DEFSOPT(
SHARED) SHARE DEFPSIST(YES)
AMQ8006: WebSphere MQ queue created.
2 : DEFINE QLOCAL(SYSTEM.PROTECTION.ERROR.QUEUE) MAXDEPTH(999999999)
MAXMSGL(4194304) DEFSOPT(S
HARED) SHARE DEFPSIST(YES)
AMQ8006: WebSphere MQ queue created.
2 MQSC commands read.
No commands have a syntax error.
All valid MQSC commands were processed.

```

7. Grant access to users to connect to the queue manager. See Example 5-7.

Example 5-7 Granting access to users to connect to the queue manager

```

C:\WMQ\bin>setmqaut -m AMS.QM -t qmgr -p storeusr -p wesbusr +connect +inq
The setmqaut command completed successfully.

```

8. Grant access to the store user to send messages. See Example 5-8.

Example 5-8 Granting access to the store user to send messages

```

C:\WMQ\bin>setmqaut -m AMS.QM -n AMS.STORE.REQUEST.Q -t queue -p storeusr +put
The setmqaut command completed successfully.

```

9. Grant access to the ESB to read messages. See Example 5-9.

Example 5-9 Granting access to the ESB to read messages

```

C:\WMQ\bin>setmqaut -m AMS.QM -n AMS.STORE.REQUEST.Q -t queue -p wesbusr +get
The setmqaut command completed successfully.

```

10. Grant browse access on the AMS system queues. See Example 5-10.

Example 5-10 Granting browse access on the AMS system queues

```

C:\WMQ\bin>setmqaut -m AMS.QM -n SYSTEM.PROTECTION.POLICY.QUEUE -t queue -p
storeusr -p wesbusr +browse
The setmqaut command completed successfully.

```

```

C:\WMQ\bin>setmqaut -m AMS.QM -n SYSTEM.PROTECTION.ERROR.QUEUE -t queue -p
storeusr -p wesbusr +browse
The setmqaut command completed successfully.

```

5.4.4 Preparing to create storeusr key database and certificates

Perform these steps to create the storeusr key database and certificates:

1. Click **Command Prompt** and click **Run as different user**, as shown in Figure 5-6.

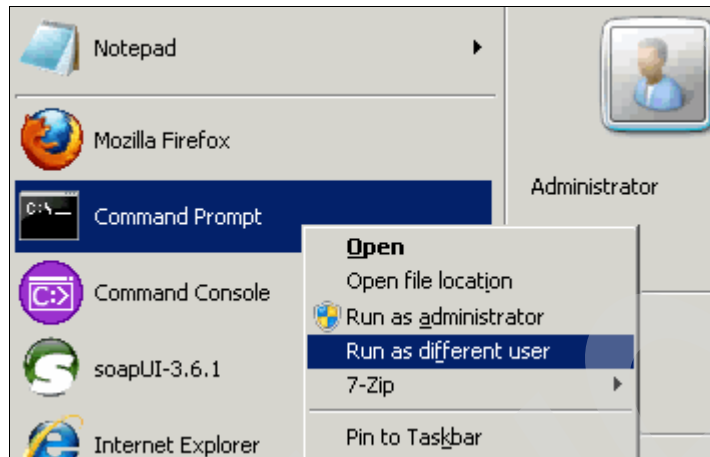


Figure 5-6 Run as different user

2. Enter the user and password on the login dialog, as shown in Figure 5-7.

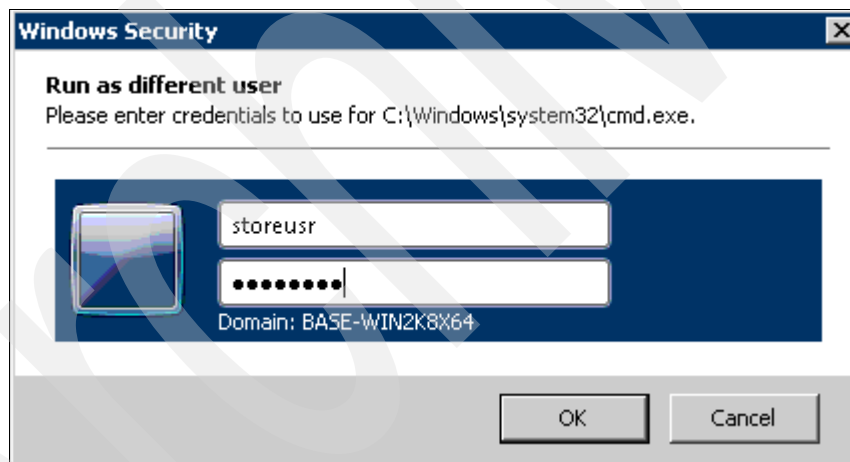


Figure 5-7 storeusr login

3. Create a command prompt as a different user. See Example 5-11.

Example 5-11 Command prompt as storeusr output

```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
```

```
C:\Users\Administrator>whoami
storeusr
```

```
C:\Users\Administrator>
```

4. Start MQ IBM Key Management by issuing the command `strmqikm`, as shown in Example 5-14 on page 217.

Example 5-12 `strmqikm`

```
C:\Users\Administrator>strmqikm
5724-H72 (C) Copyright IBM Corp. 1994, 2009. ALL RIGHTS RESERVED.

C:\Users\Administrator>
```

MQ IBM Key Management starts, as shown in Figure 5-8.

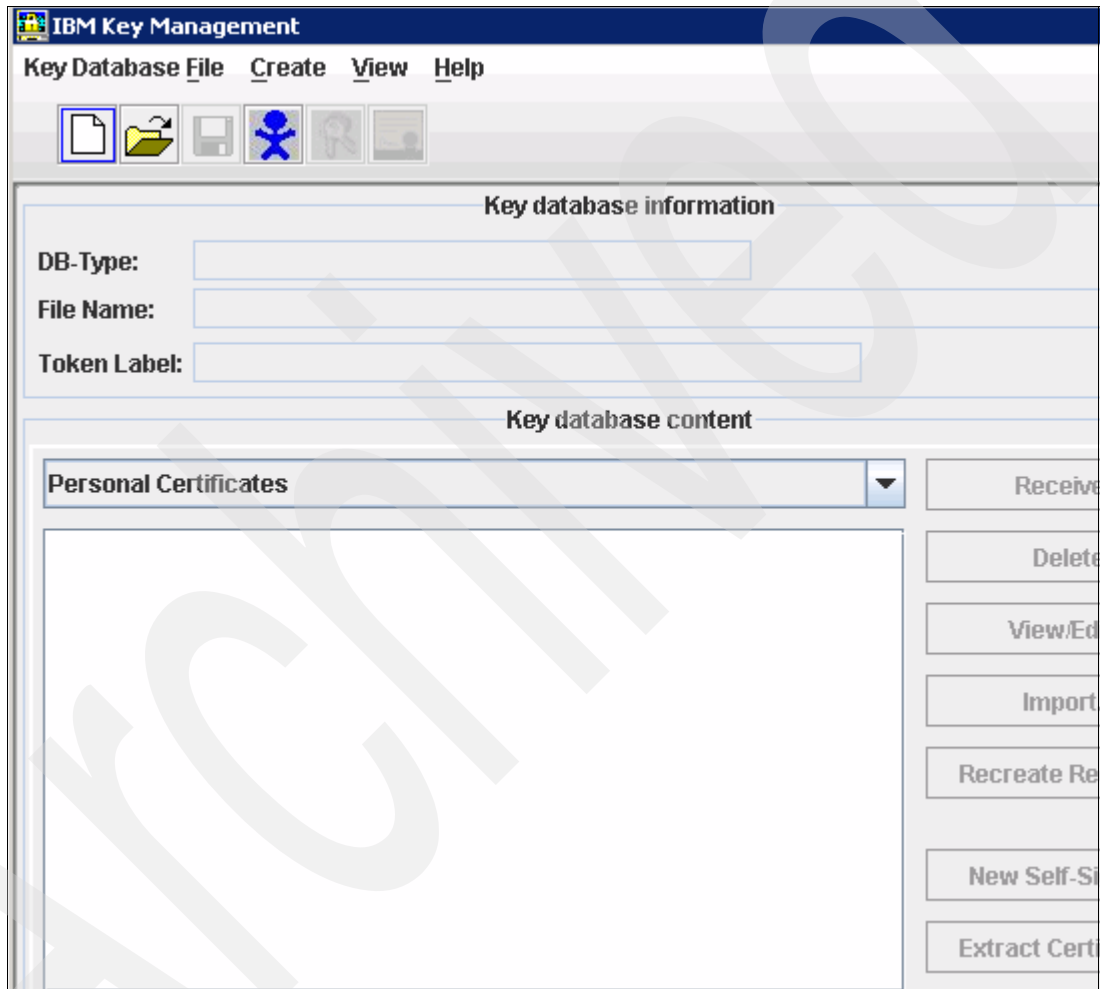


Figure 5-8 IBM Key Management

5. Create a new key database file by using the menu **Key Database File** → **New**, as shown in Figure 5-9 on page 214.

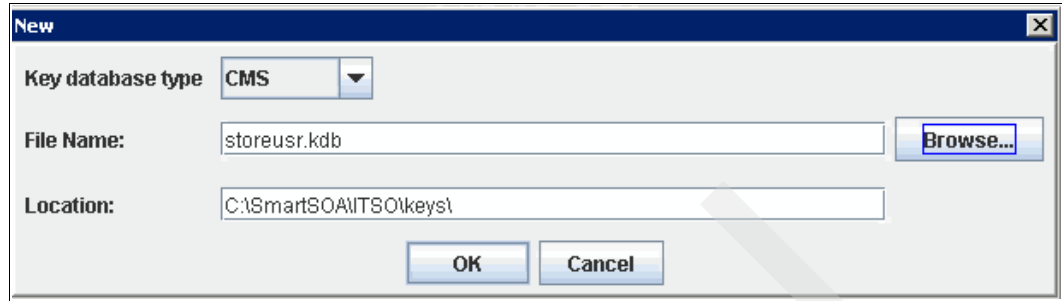


Figure 5-9 New CMS key database for storeusr

6. Set the password for the key database file. Remember to enable **Stash the password to a file**. See Figure 5-10.

Password conventions: For a production environment, make sure that you follow your security policy standards for passwords.

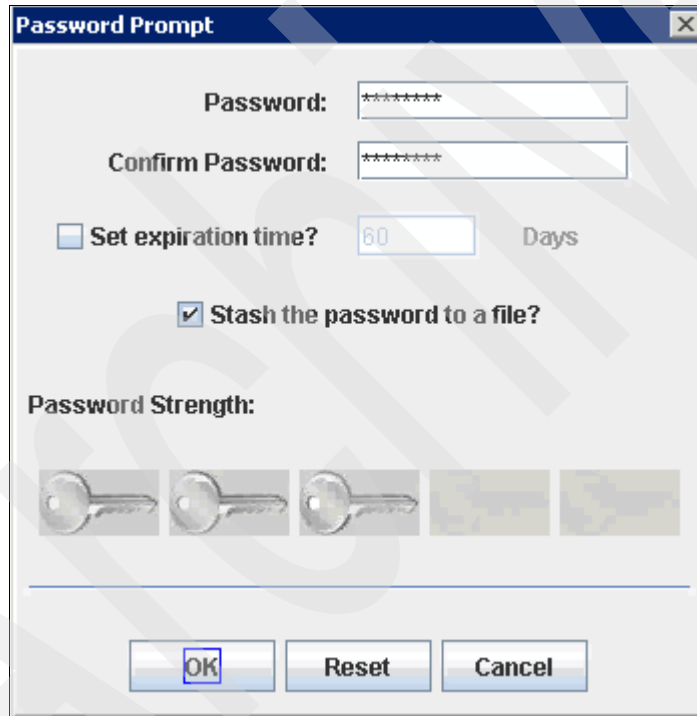


Figure 5-10 Password prompt for the storeusr.kdb file

A message dialog confirms the name of the stash file, as shown in Figure 5-11 on page 215.



Figure 5-11 Stash file information message

7. Change the key database content combination box to show the Personal Certificate List and create a new self-signed certificate, as shown in Figure 5-12.

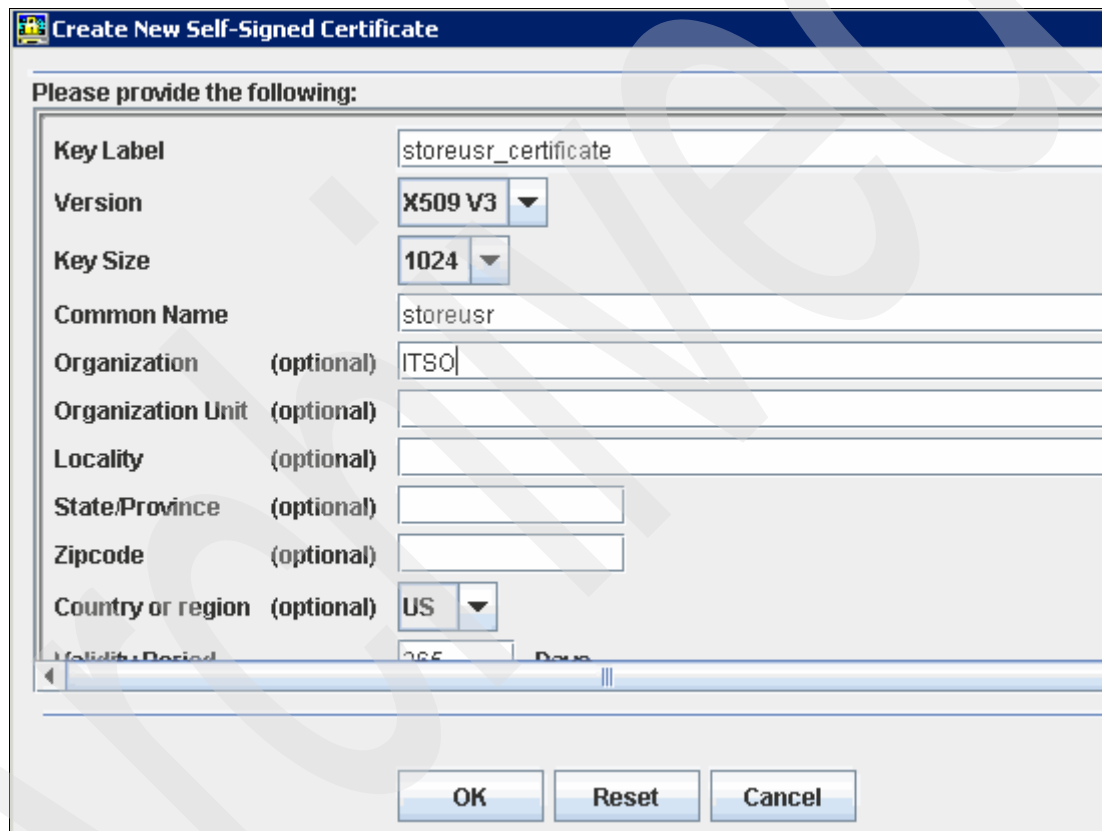


Figure 5-12 New Self-Signed Certificate for the storeusr

Now, your personal certificate list looks like Figure 5-13 on page 216.

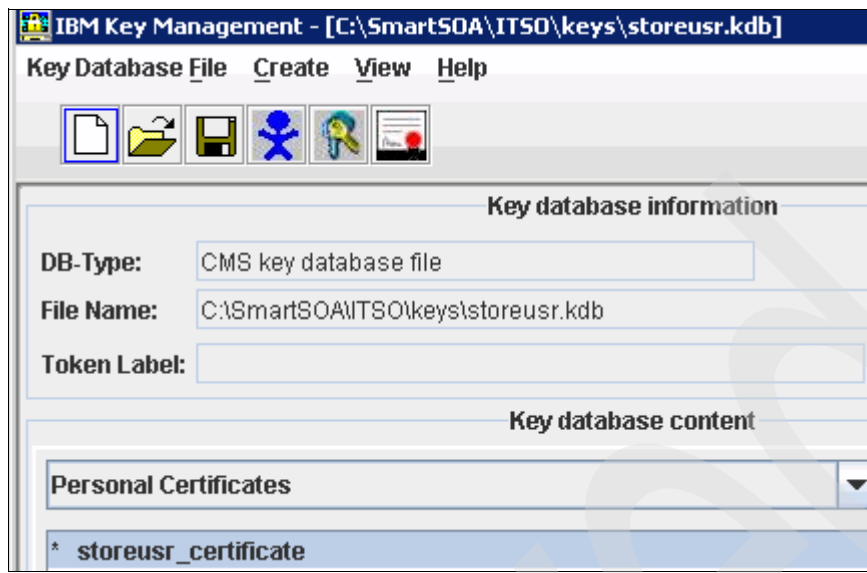


Figure 5-13 CMS key database personal certificate list updated

5.4.5 Creating a wesbusr key database and certificates

Perform the following steps to create a wesbusr key database and certificates:

1. Click **Command Prompt** and select **Run as different user**. See Figure 5-14.

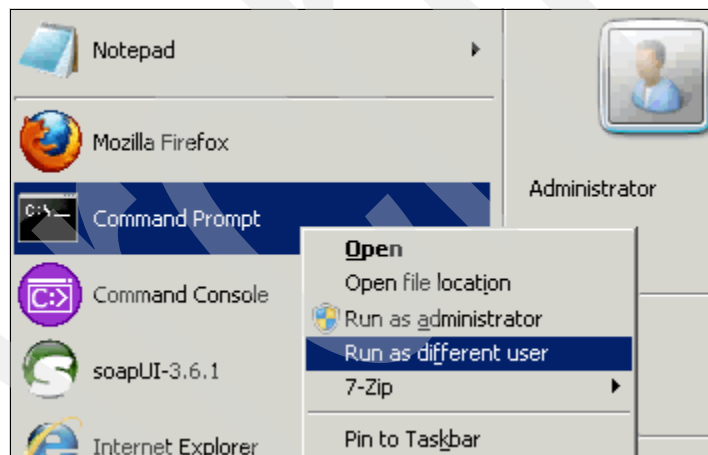


Figure 5-14 Run Command Prompt as different user

2. Log in as wesbusr. See Figure 5-15 on page 217.

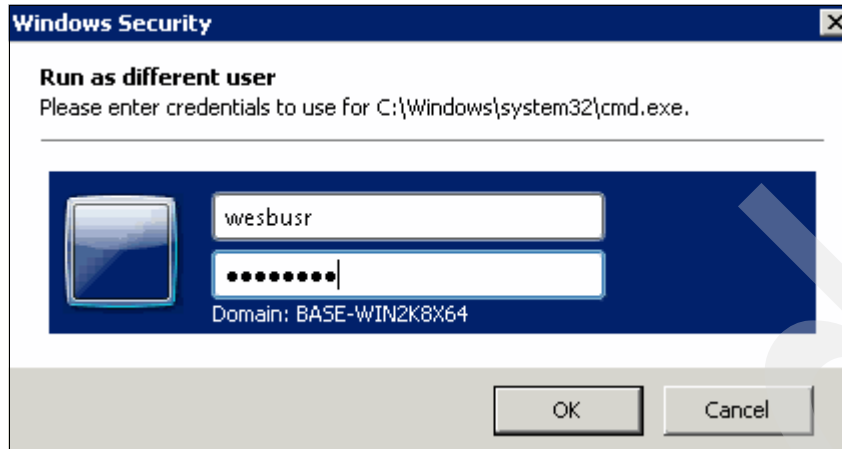


Figure 5-15 Log in as wesbusr

3. Example 5-13 shows the command prompt as another user.

Example 5-13 Command prompt as wesbusr output

```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
```

```
C:\Users\Administrator>whoami
wesbusr
```

```
C:\Users\Administrator>
```

4. Start MQ IBM Key Management by issuing the command **strmqikm**, as shown in Example 5-14.

Example 5-14 strmqikm

```
C:\Users\Administrator>strmqikm
5724-H72 (C) Copyright IBM Corp. 1994, 2009. ALL RIGHTS RESERVED.
```

```
C:\Users\Administrator>
```

The MQ IBM Key Management starts, as shown in Figure 5-16 on page 218.

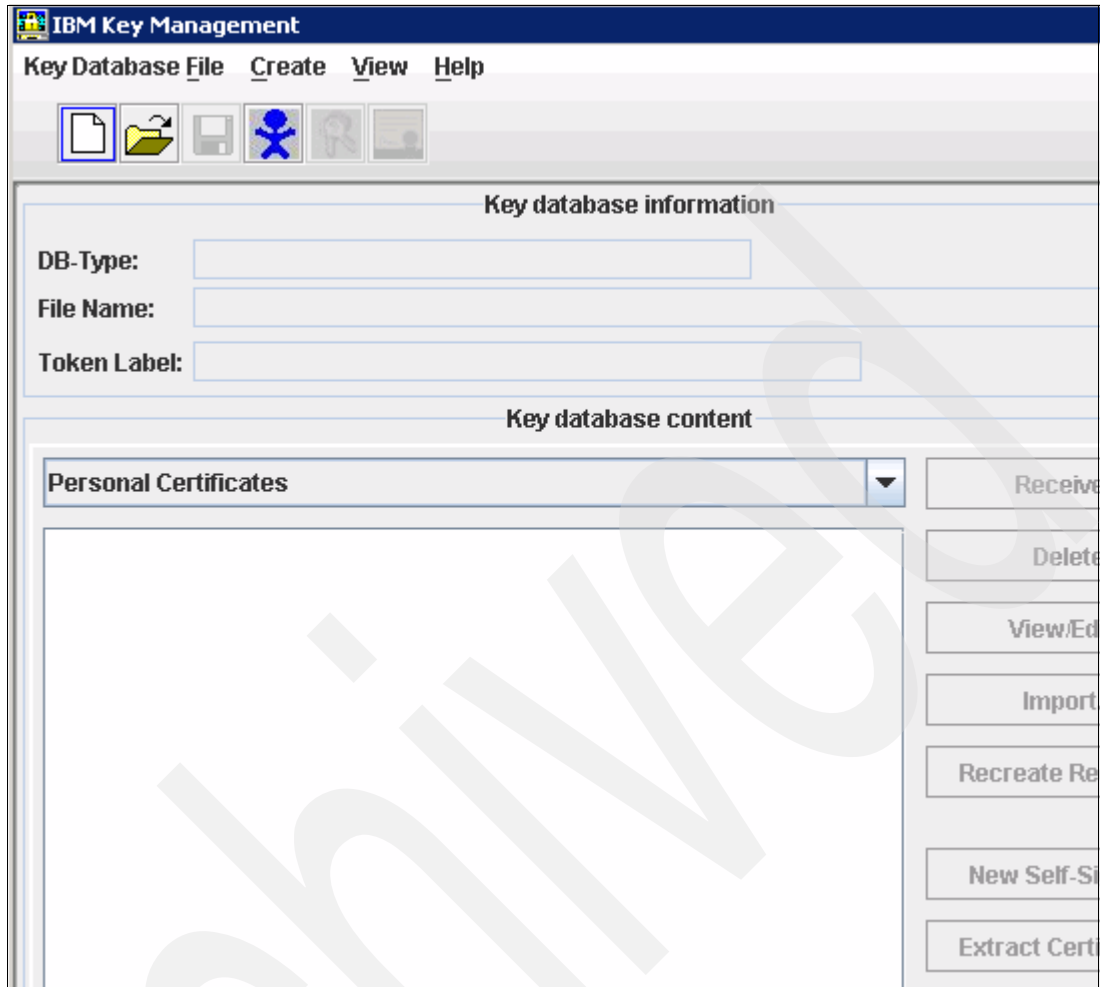


Figure 5-16 IBM Key Management

5. Create a new JKS key database, as shown in Figure 5-17.

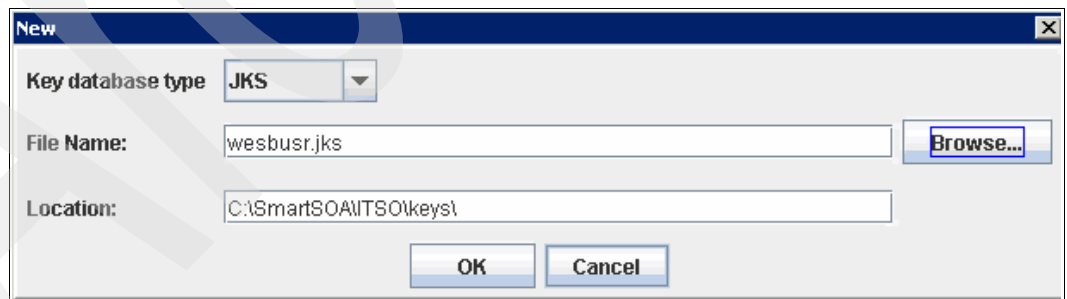


Figure 5-17 Create a new JKS key database

6. Set the JKS key database password. See Figure 5-18 on page 219.

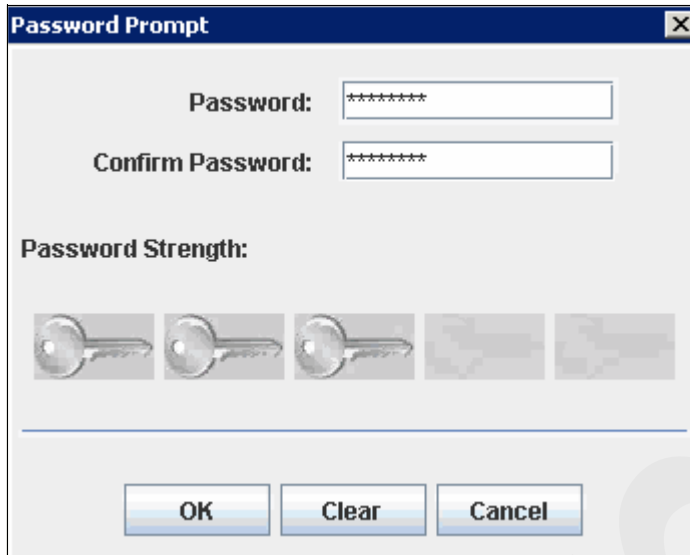


Figure 5-18 Key database password set

7. Change the key database content combination box to **Personal Certificates**, as shown in Figure 5-19.



Figure 5-19 JKS key database Personal Certificates list

8. Create a new self-signed certificate. See Figure 5-20 on page 220.

Figure 5-20 Create New Self-Signed Certificate window

After creating the self-signed certificate, your list of personal certificates looks like Figure 5-21.

Figure 5-21 JKS key database personal certificates list updated

5.4.6 Exchanging certificates

To make the message encryption work as planned, the message sender and receiver must “know” each other by exchanging certificates that have been previously created.

Because we are using the same machine to make this work, we only need to import the certificates from one key database file to the other key database file:

1. Import the JKS certificate into the storeusr.kdb database file. Open IBM Key Management and the storeusr.kdb key database file and change to the personal certificates list.
2. Import the CMS certificate that was created by the storeusr to the JKS key database that was created using wesbusr, as shown in Figure 5-22 on page 221.

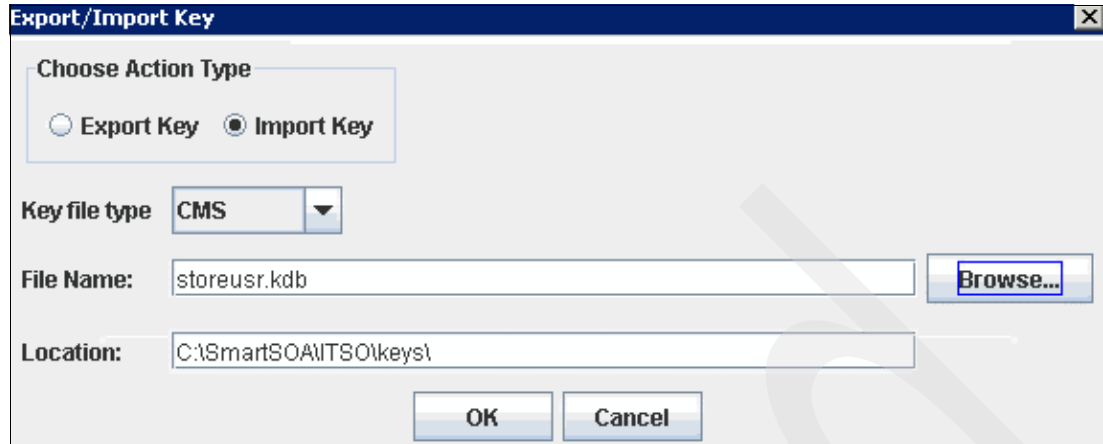


Figure 5-22 Import certificate from storeusr.kdb

3. Type the password for the storeusr.kdb key database file. The dialog looks like Figure 5-23.

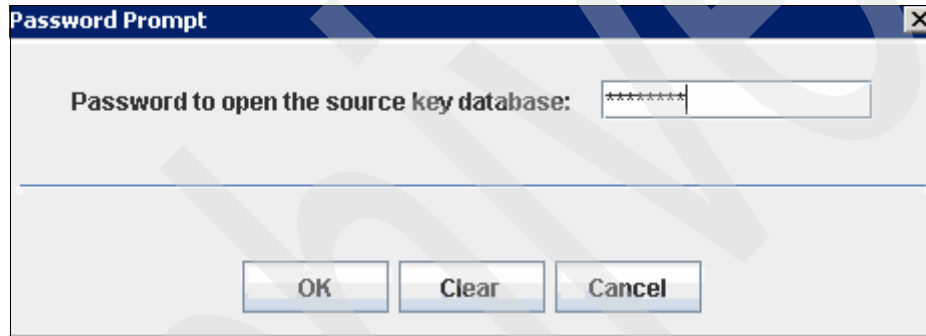


Figure 5-23 Password dialog for importing the CMS certificate

4. Choose the certificate that you want to import. In this example, we import **storeusr_certificate**, as shown in Figure 5-24.

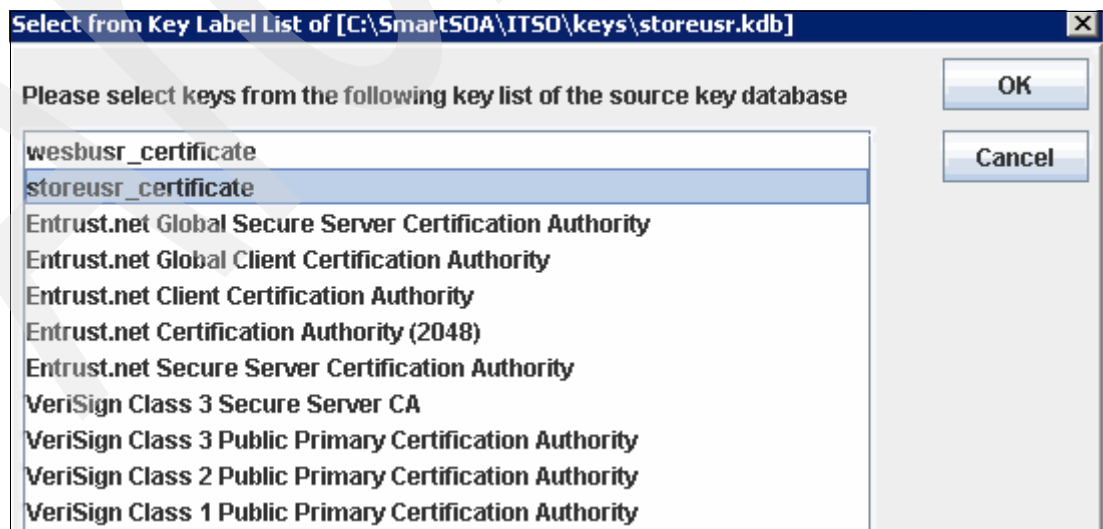


Figure 5-24 storeusr.kdb certificate list

5. You will be prompted to a window where you can change the label. It is optional, so for this example, we use the same name on both databases. (Figure 5-25).

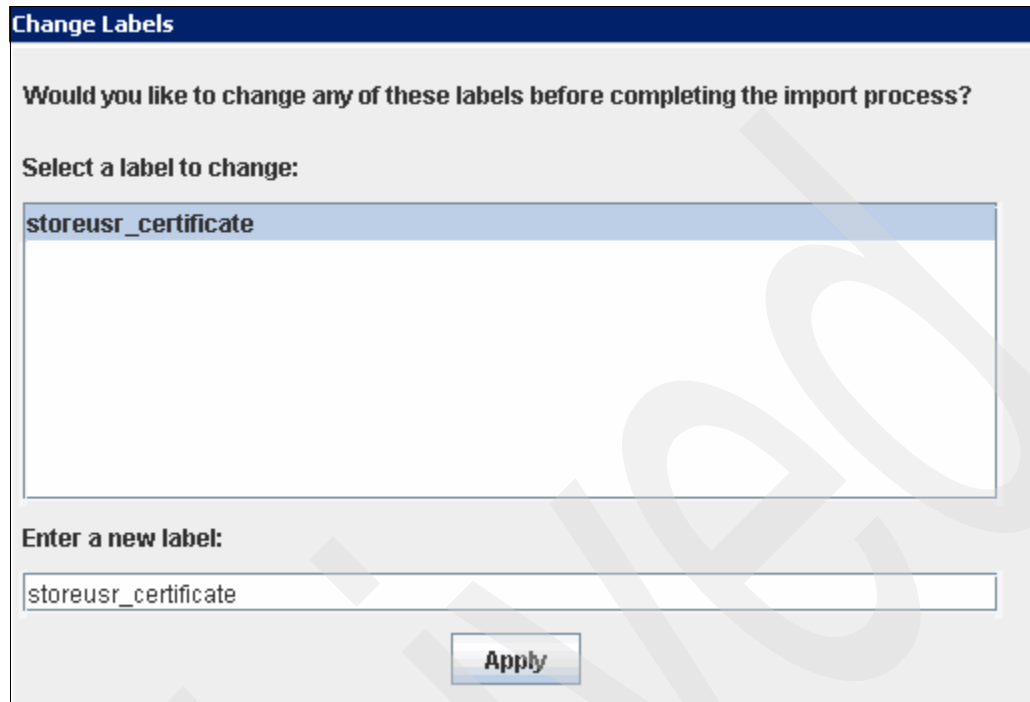


Figure 5-25 Import key Change Labels window

After importing the certificate, the `wesbusr.jks` personal certificates list looks like Figure 5-26.

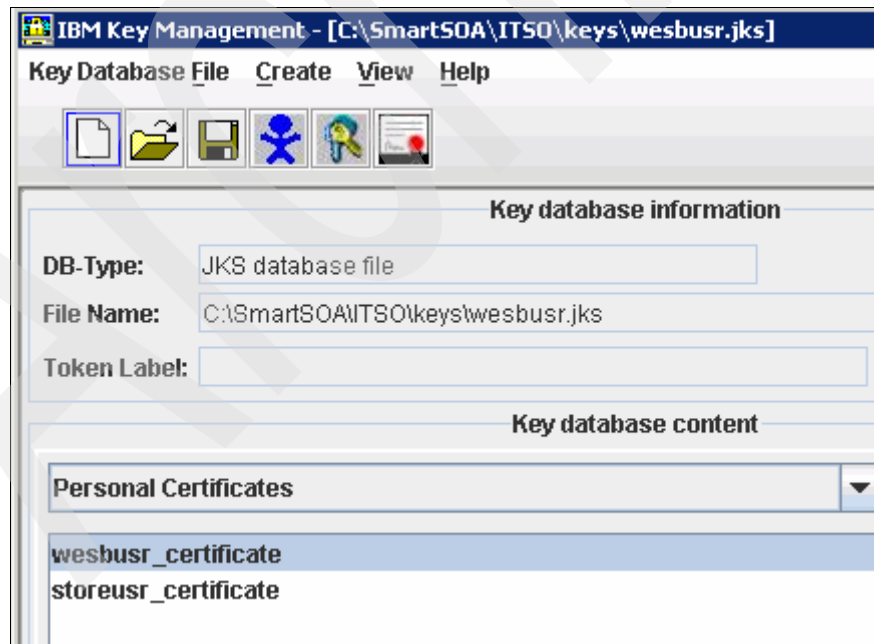


Figure 5-26 `wesbusr.jks` Personal Certificates list updated

5.4.7 Implementing the solution

Before moving to the implementation, make sure that the existing infrastructure allows you to exchange messages between the two users:

1. Open a command prompt as the storeusr and put a test message on the queue using the **amqsput** sample MQ program (Example 5-15).

Example 5-15 Sending a test message to the queue

```
C:\Users\Administrator>whoami
storeusr

C:\Users\Administrator>amqsput AMS.STORE.REQUEST.Q AMS.QM
Sample AMQSPUTO start
target queue is AMS.STORE.REQUEST.Q
TEST

Sample AMQSPUTO end
```

2. Open another command prompt as the wesbusr user and get the message from the queue using the **amqsget** sample MQ program (Example 5-16).

Example 5-16 Getting the test message from the queue

```
C:\Users\Administrator>whoami
wesbusr

C:\Users\Administrator>amqsget AMS.STORE.REQUEST.Q AMS.QM
Sample AMQSGETO start
message <TEST>
no more messages
Sample AMQSGETO end

C:\Users\Administrator>
```

3. Open a third command prompt as the Administrator user and try to both put and get a message from the queue (Example 5-17 and Example 5-18 on page 224).

Example 5-17 Sending a message as Administrator

```
C:\Users\Administrator>amqsput AMS.STORE.REQUEST.Q AMS.QM
Sample AMQSPUTO start
target queue is AMS.STORE.REQUEST.Q
TESTING 123

Sample AMQSPUTO end

C:\Users\Administrator>
```

Example 5-18 Getting a message as Administrator

```
C:\Users\Administrator>whoami
Administrator

C:\Users\Administrator>AMQSGET AMS.STORE.REQUEST.Q AMS.QM
Sample AMQSGET0 start
message <TESTING 123>
no more messages
Sample AMQSGET0 end
```

Creating a Java client

We suggest that you create two simple Java client programs to put and get messages from the queue before you move on to the WebSphere ESB implementation, to make sure that you have done all the preparation tasks properly. Example 5-19 shows sample code for put messages.

Example 5-19 MQPut.java

```
import java.io.IOException;

import com.ibm.mq.MQException;
import com.ibm.mq.MQMessage;
import com.ibm.mq.MQPutMessageOptions;
import com.ibm.mq.MQQueue;
import com.ibm.mq.MQQueueManager;
import com.ibm.mq.constants.MQConstants;

/**
 * Simple example program
 */
public class MQPut {

    private static final String qManager = "AMS.QM";
    private static final String qName = "AMS.STORE.REQUEST.Q";

    public static void main(String args[]) {
        try {
            System.out.println("Connecting to the queue manager");
            MQQueueManager qMgr = new MQQueueManager(qManager);
            int openOptions = MQConstants.MQOO_OUTPUT;
            System.out.println("Accessing the queue");
            MQQueue queue = qMgr.accessQueue(qName, openOptions);
            MQMessage msg = new MQMessage();
            System.out.println("Writing a simple message");
            msg.writeUTF("Can you read this message?");
            MQPutMessageOptions pmo = new MQPutMessageOptions();
            queue.put(msg, pmo);
            queue.close();
            qMgr.disconnect();
            System.out.println("Done. Message: 'Can you read this message?' has been sent.");
        } catch (MQException ex) {
            System.out.println("A WebSphere MQ Error occurred : Completion Code " + ex.completionCode
                + " Reason Code " + ex.reasonCode);
        }
    }
}
```

```

        ex.printStackTrace();
    } catch (IOException e) {
e.printStackTrace();
    }
    return;
}
}

```

Example 5-20 shows a sample Java client to get messages from the queue.

Example 5-20 MQGet.java

```

import com.ibm.mq.MQEnvironment;
import com.ibm.mq.MQException;
import com.ibm.mq.MQGetMessageOptions;
import com.ibm.mq.MQMessage;
import com.ibm.mq.MQQueue;
import com.ibm.mq.MQQueueManager;
import com.ibm.mq.constants.MQConstants;

/**
 * Simple example program
 */
public class MQGet {

    public static void init() {
        MQEnvironment.hostname = "9.42.171.212";
        MQEnvironment.channel = "AMS.CHANNEL";
        MQEnvironment.properties.put(MQConstants.TRANSPORT_PROPERTY,
            MQConstants.TRANSPORT_MQSERIES);
    }

    private static final String qManager = "AMS.QM";
    private static final String qName = "AMS.STORE.REQUEST.Q";

    public static void main(String args[]) {
        try {
            init();
            System.out.println("Connecting to the queue manager");
            MQQueueManager qMgr = new MQQueueManager(qManager);
            int openOptions = MQConstants.MQOO_INPUT_AS_Q_DEF;
            System.out.println("Accessing the queue");
            MQQueue queue = qMgr.accessQueue(qName, openOptions);
            MQMessage rcvMessage = new MQMessage();
            MQGetMessageOptions gmo = new MQGetMessageOptions();
            queue.get(rcvMessage, gmo);
            String msgText = rcvMessage.readUTF();
            System.out.println("The message is: " + msgText);
            queue.close();
            qMgr.disconnect();
            System.out.println("MQGet done.");
        } catch (MQException ex) {
            ex.printStackTrace();
        } catch (Exception ex) {
            System.out.println("An IOException occurred whilst writing to the message
buffer: " + ex);

```

```
        ex.printStackTrace();
    }
    return;
}
}
```

Examples: We have suppressed the comments and logs to fit better on the page.

To run the programs, you need to make sure that you have the following jars in your classpath environment variable or Java virtual machine (JVM) argument:

- ▶ com.ibm.mq.jar
- ▶ com.ibm.com.jmqi.jar
- ▶ com.ibm.mqjms.jar

Important: At this point, we still do not have WebSphere MQ Advanced Message Security interceptors. After we create the WebSphere MQ Advanced Message Security protection policy, an IBM JVM is required.

The output for the MQPut program looks like Example 5-21.

Example 5-21 MQPut output

```
Connecting to the queue manager
Accessing the queue
Writing a simple message
Done. Message: 'Can you read this message?' has been sent.
```

The output for the MQGet program looks like Example 5-22.

Example 5-22 MQGet program output

```
Connecting to the queue manager
Accessing the queue
The message is: Can you read this message?
MQGet done.
```

Creating the protection policy

The protection policy is the main configuration for the Message Privacy Enforcement pattern. There are more options than we present in this chapter. Refer to the documentation to better understand the possibilities that WebSphere MQ Advanced Message Security can provide.

In Example 5-23, we create a protection policy for the AMS.STORE.REQUEST.Q queue. The protection policy guarantees that storeusr messages will be encrypted using the certificates that were created previously and that only wesbusr is authorized to decrypt these messages.

Example 5-23 Creating the protection policy

```
C:\WMQAMS\bin>setmqspl -m AMS.QM -p AMS.STORE.REQUEST.Q -s SHA1 -a
"CN=storeusr,0=ITSO,C=US" -e AES256 -r "CN=wesbusr,0=ITSO,C=US"

C:\WMQAMS\bin>
```

Table 5-1 on page 227 explains each argument that was used while creating the protection policy.

Table 5-1 *setmqspl flags*

Command flag	Explanation
-m	Queue manager name
-p	Policy name
-s	Digital signature algorithm
-a	Signature distinguished name (DN), which is validated during the message retrieval (only messages that are signed by a user with a DN provided are accepted during retrieval)
-e	Digital encryption algorithm
-r	Message recipient's DN (a certificate of a DN provided is used to encrypt a given message)

For more information: See `setmqspl` at this website:

http://publib.boulder.ibm.com/infocenter/mqams/v7r0m1/index.jsp?topic=/com.ibm.mqese.doc/reference/reference_setmqspl.htm

Enabling MQ interceptors on the server side

No changes are needed in the applications that make use of WebSphere MQ queues to encrypt messages end-to-end. The change is done in the infrastructure via interceptors that read configurations and guarantee that the protection policy is assured.

On the server side, we must enable the server interceptor, as demonstrated in Example 5-24.

Example 5-24 Enabling server interceptor

```
C:\Users\Administrator>cfgmqms -enable -server AMS.QM
```

```
DRQDT3052I The IBM WebSphere MQ Advanced Message Security server interceptor has been enabled successfully.
```

```
C:\Users\Administrator>
```

To make the protection policy active and properly implemented for Java applications, such as WebSphere Enterprise Service Bus flows, it is mandatory for the Java interceptor to be enabled. To enable the Java interceptor, issue the command that is shown in Example 5-25.

Example 5-25 Enabling the Java interceptor

```
C:\WMQAMS\bin>cfgmqms.bat -enable -java
```

```
DRQDT3028 The IBM WebSphere MQ Advanced Message Security Java interceptor has been enabled successfully.
```

Configuring WebSphere Enterprise Service Bus

Although you do not need to make changes to your application, you still have to set up your environment. Now that the protection policy is created and enabled, what is defined in the protection policy cannot be bypassed. For instance, if we try to put a message in the queue with a user that has authorization to put a message in the queue but is not defined in the protection policy, MQ will block the attempt with a RC2063 (Example 5-26 on page 228).

Example 5-26 AMS protection policy in action

```
C:\Users\Administrator>dspmqaout -m AMS.QM -t queue -n AMS.Q -p otherusr
Entity otherusr has the following authorizations for object AMS.Q:
    put
```

```
C:\SmartSOA\mqsample>mqrc 2063
```

```
2063 0x0000080f MQRC_SECURITY_ERROR
```

```
C:\SmartSOA\mqsample>whoami
otherusr
```

```
C:\SmartSOA\mqsample>c:\wpsv75\java\bin\java MQPut
Connecting to queue manager: AMS.QM
Accessing queue: AMS.Q
MQJE001: Completion Code '2', Reason '2063'.
A WebSphere MQ Error occurred : Completion Code 2 Reason Code 2063
com.ibm.mq.MQException: MQJE001: Completion Code '2', Reason '2063'.
    at com.ibm.mq.MQDestination.open(MQDestination.java:328)
    at com.ibm.mq.MQQueue.<init>(MQQueue.java:257)
    at com.ibm.mq.MQQueueManager.accessQueue(MQQueueManager.java:2750)
    at com.ibm.mq.MQQueueManager.accessQueue(MQQueueManager.java:2778)
    at MQPut.main(MQPut.java:52)
```

```
C:\SmartSOA\mqsample>
```

The WebSphere MQ Advanced Message Security Java interceptor requires that strong encryption is enabled for the Java Runtime Environment (JRE). To enable strong encryption, replace the restricted Java Cryptography Extension (JCE) policy files in the `$JAVA_HOME/jre/lib/security` directory with unrestricted files. You can obtain unrestricted jurisdiction policy files (IBM Software Development Kit (SDK) Policy files) from the following locations:

- ▶ IBM SDK Policy files for Java 1.4.2:
<https://www.ibm.com/developerworks/java/jdk/security/142/>
- ▶ IBM SDK Policy files for Java 5.0:
<http://www.ibm.com/developerworks/java/jdk/security/50/>
- ▶ IBM SDK Policy files for Java 6.0:
<http://www.ibm.com/developerworks/java/jdk/security/60/>

Important: Without these files, you will see a `java.security.InvalidKeyException: Illegal key size` in your error logs.

To provide transparent cryptographic protection to WebSphere MQ applications, IBM WebSphere MQ Advanced Message Security uses the keystore file, where public and private keys are stored.

In WebSphere MQ Advanced Message Security, users and applications are represented by public key infrastructures (PKIs). This type of identity is used for signing and encrypting messages. PKI identity is represented by the subject's distinguished name (DN) field in a certificate that is associated with signed and encrypted messages. To authenticate this identity, the user or application must have access to the keystore file where certificates and the associated private and public keys are stored.

The keystore configuration file (by default, it is `keystore.conf`) contains information about the location of the key that is stored and the label of the certificate to use. Each user or service using WebSphere MQ Advanced Message Security needs the configuration file. This configuration file points to a keystore file, which contains the private key of a user or service along with its corresponding public key certificate and public keys or parties with which the specific user or service communicates. The `.kdb` configuration file is used by WebSphere MQ Advanced Message Security (MQ AMS) native interceptors (Server and Client), while configuration files with `.jceks` and `.jks` extensions are used by the Java interceptor. The file must be created by the users.

For more information: To learn more about the keystore configuration file, follow the link:
http://publib.boulder.ibm.com/infocenter/mqams/v7r0m1/index.jsp?topic=/com.ibm.mqese.doc/configuring/configuring_mapping.htm

For the `storeusr`, we use a CMS Key DB file. Example 5-27 shows a simple configuration file.

Example 5-27 CMS Key DB config file

```
cms.keystore=C:\\SmartSOA\\keys\\storeusr
cms.certificate=storeusr_certificate
```

Microsoft Windows: On Microsoft Windows, you must specify the path to the keystore file platforms by using slash (/) or two backslashes (\\).

Important: The path to the keystore file must not have an extension.

The IBM WebSphere Enterprise Service Bus process ID is `wesbusr`. Because it is a Java client application, we must use a supported Key DB file. In this case, the chosen type is JKS. Example 5-28 shows a configuration file for the AMS interceptors.

Example 5-28 JKS Key DB configuration file

```
jks.keystore=C:\\SmartSOA\\keys\\wesbusr
jks.certificate=wesbusr_certificate
jks.encrypted=no
jks.keystore_pass=itso4you
jks.key_pass=itso4you
jks.provider=IBMJCE
```

To protect your password, refer to the following link:

http://publib.boulder.ibm.com/infocenter/mqams/v7r0m1/topic/com.ibm.mqese.doc/configuring/configuring_passwordProtection.htm

Before you move to your application server-deployed application configurations and tests, make sure that you have followed all steps by testing with the Java programs that were created previously. However, the test will not work properly without additional JVM arguments. We have created two configuration files: one file for the user that will send messages to the queue (`storeusr`) and the other file for the user that will get messages from the queue (`wesbusr`). To make it easier for the testing, create two `.bat` files to help with the additional arguments.

The `storeusr` configuration file will be used when putting messages, so create an `mqput.bat` file, as shown in Example 5-29 on page 230.

Example 5-29 MQPut.bat

```
set JAVA_HOME=C:/WPSv75/WebSphere/AppServer/java
Set MQS_KEYSTORE_CONF=C:/SmartSOA/conf/storeusr.conf

%JAVA_HOME%\bin\javac MQPut.java
%JAVA_HOME%\bin\java -cp %CLASSPATH%;C:\WMQAMS\classes\com.ibm.mq.ese.jar
-DMQS_KEYSTORE_CONF=C:/SmartSOA/conf/storeusr.conf
-Dmq.dir.config=c:/WMQAMS/config -Dmq.dir.trace=c:/SmartSOA/logs/put
-Dmq.dir.log=c:/SmartSOA/logs/put -DMQS_NAME=MQPut MQPut
```

For more information: MQS_NAME, mqs.dir.config, mqs.dir.trace, and mqs.dir.log are optional arguments, but they help with troubleshooting. For additional information, refer to the following page:

http://publib.boulder.ibm.com/infocenter/mqams/v7r0m1/topic/com.ibm.mq.ese.doc/troubleshooting/troubleshooting_tracesandroutingfilesforjavaapplications.htm

The wesbusr configuration file will be used when getting messages. Create a .bat file to help you, as shown in Example 5-30.

Example 5-30 MQGet.bat

```
set JAVA_HOME=C:/WPSv75/WebSphere/AppServer/java
set MQS_KEYSTORE_CONF=C:/SmartSOA/conf/wesbusr.conf

%JAVA_HOME%\bin\javac MQGet.java
%JAVA_HOME%\bin\java -cp %CLASSPATH%;C:\SmartSOA\mqsample\com.ibm.mq.ese.jar
-Dmq.dir.config=c:/WMQAMS/config -Dmq.dir.trace=c:/SmartSOA/logs
-Dmq.dir.log=c:/SmartSOA/logs -DMQS_NAME=MQGet
-DMQS_KEYSTORE_CONF=C:/SmartSOA/conf/wesbusr.conf MQGet
```

com.ibm.mq.ese.jar file: Notice that com.ibm.mq.ese.jar has been added to the CLASSPATH. Now, whenever a Java application starts, MQ .jar files, together with the com.ibm.mq.ese.jar file, will be loaded dynamically, which enables the WebSphere MQ Advanced Message Security Java interceptor.

WebSphere MQ Advanced Message Security provides the com.ibm.mq.ese.jar file. You can find it at %WMQAMS%\classes.

Example 5-31 shows a sample output for the MQput.bat.

Example 5-31 MQput.bat output

```
C:\SmartSOA\mqsample>mqput

C:\SmartSOA\mqsample>set JAVA_HOME=C:/WPSv75/WebSphere/AppServer/java

C:\SmartSOA\mqsample>Set MQS_KEYSTORE_CONF=C:/SmartSOA/conf/storeusr.conf

C:\SmartSOA\mqsample>C:/WPSv75/WebSphere/AppServer/java/bin/javac MQPut.java

C:\SmartSOA\mqsample>C:/WPSv75/WebSphere/AppServer/java/bin/java -cp C:\WMQ\Java\lib\com.ibm.mqjms.jar;C:\WMQ\Java\lib\com.ibm.mq.jar;.;C:\WPSv75\WEBSPH~1\APPSE
```

```

R~1\db2\java\db2java.zip;C:\WPSv75\WEBSPH~1\APPSE~1\db2\java\db2jcc.jar;C:\WPSv
75\WEBSPH~1\APPSE~1\db2\java\sqlj.zip;C:\WPSv75\WEBSPH~1\APPSE~1\db2\java\db2j
cc_license_cu.jar;C:\WPSv75\WEBSPH~1\APPSE~1\db2\bin;C:\WPSv75\WEBSPH~1\APPSE~
1\db2\java\common.jar;C:\WMQAMS\classes\com.ibm.mq.ese.jar -DMQS_KEYSTORE_CONF=C
:/SmartSOA/conf/storeusr.conf -Dmq.dir.config=c:/WMQAMS/config -Dmq.dir.trace=
c:/SmartSOA/logs/put -Dmq.dir.log=c:/SmartSOA/logs/put -DMQS_NAME=MQPut MQPut
Connecting to queue manager: AMS.QM
Accessing queue: AMS.STORE.REQUEST.Q
Define a simple WebSphere MQ Message ...
    // ... and write some text in UTF8 format
Sending a message...
Closing the queue
Disconnecting from the Queue Manager
Done!

```

```
C:\SmartSOA\mqsample>
```

Example 5-32 shows a sample output for the MQGet.bat.

Example 5-32 MQGet.bat output

```
C:\SmartSOA\mqsample>mqget
```

```
C:\SmartSOA\mqsample>set JAVA_HOME=C:/WPSv75/WebSphere/AppServer/java
```

```
C:\SmartSOA\mqsample>set MQS_KEYSTORE_CONF=C:/SmartSOA/conf/wesbusr.conf
```

```
C:\SmartSOA\mqsample>C:/WPSv75/WebSphere/AppServer/java/bin/javac MQGet.java
```

Note: MQGet.java uses unchecked or unsafe operations.

Note: Recompile with -Xlint:unchecked for details.

```

C:\SmartSOA\mqsample>C:/WPSv75/WebSphere/AppServer/java/bin/java -cp C:\WMQ\Java
\lib\com.ibm.mqjms.jar;C:\WMQ\Java\lib\com.ibm.mq.jar;.;C:\WPSv75\WEBSPH~1\APPSE
R~1\db2\java\db2java.zip;C:\WPSv75\WEBSPH~1\APPSE~1\db2\java\db2jcc.jar;C:\WPSv
75\WEBSPH~1\APPSE~1\db2\java\sqlj.zip;C:\WPSv75\WEBSPH~1\APPSE~1\db2\java\db2j
cc_license_cu.jar;C:\WPSv75\WEBSPH~1\APPSE~1\db2\bin;C:\WPSv75\WEBSPH~1\APPSE~
1\db2\java\common.jar;C:\SmartSOA\mqsample\com.ibm.mq.ese.jar -Dmq.dir.config=c
:/WMQAMS/config -Dmq.dir.trace=c:/SmartSOA/logs -Dmq.dir.log=c:/SmartSOA/logs
-DMQS_NAME=MQGet -DMQS_KEYSTORE_CONF=C:/SmartSOA/conf/wesbusr.conf MQGet
Connecting to the queue manager
Accessing the queue
The message is: Hello, World!
MQGet done.
C:\SmartSOA\mqsample>

```

The following steps show the configuration for the WebSphere ESB configurations:

1. Add com.ibm.mq.ese.jar to the MQ Resource Adapter.

Copy the WebSphere MQ Advanced Message Security jar file (com.ibm.mq.ese.jar) to a location where WebSphere MQ Resource Adapter has been installed for your application server (this location, among other files, will contain the com.ibm.mq.jmqi.jar file).

WebSphere MQ Advanced Message Security jar file: The WebSphere MQ Advanced Message Security jar file (`com.ibm.mq.ese.jar`) is installed with the WebSphere MQ Advanced Message Security Java package. It is located in the `<MQAMS_data_dir>/classes` directory.

2. Add WebSphere MQ Advanced Message Security (MQ AMS) arguments to the JVM properties.

For your WebSphere ESB application server, set a new JVM system property named `MQS_KEYSTORE_CONF` that will tell WebSphere MQ Advanced Message Security where to locate the keystore configuration file:

- a. Log in to Integrated Solutions Console.
- b. Select **Servers** → **Server Types** → **WebSphere application servers** and choose your application server.
- c. Select **Server Infrastructure** → **Java and Process Management** and click **Process definition**.
- d. In Additional Properties, select **Java Virtual Machine**.
- e. In Additional Properties, select **Custom properties** and click **New** to add a new system property.
- f. Type `MQS_KEYSTORE_CONF` as the name.
- g. Provide the location of the keystore configuration file in the Value field.
- h. Click **Apply** and click **Save**.
- i. Restart the WebSphere Enterprise Service Bus application server.

Important: Make sure that your connection factory or activation specification in WebSphere Application Server is configured in the following ways:

- ▶ WebSphere MQ Resource Adapter is specified as a JMS resource provider.
- ▶ Client is specified as a transport mode.

5.5 Summary

ITSO Enterprise has implemented the requirement for the encryption of messages containing credit card information end-to-end. No application code changes have been done. What is required is to prepare the environment. In summary, ITSO Enterprise performed these steps to implement WebSphere MQ Advanced Message Security:

- ▶ Install WebSphere MQ Advanced Message Security
- ▶ Create and exchange certificates
- ▶ Create a protection policy
- ▶ Enable WebSphere MQ Advanced Message Security interceptors
- ▶ Create the required configuration files for WebSphere MQ Advanced Message Security
- ▶ Add the WebSphere MQ Advanced Message Security interceptors' library to the classpath

ITSO Enterprise business requirements have been addressed by implementing a Smart SOA solution using the existing installed IT resources.

Enterprise Messaging and File Backbone pattern

In this chapter, we introduce the Enterprise Messaging and File Backbone pattern for ITSO Enterprise. A solution that is based on this pattern will be implemented in ITSO Enterprise using IBM WebSphere MQ File Transfer Edition and IBM WebSphere MQ.

After this pattern is implemented for ITSO Enterprise, the company will achieve the following benefits:

- ▶ Audit control of file transfer
- ▶ Secure communication
- ▶ Centralized administration
- ▶ End-to-end automation
- ▶ Checkpoint and restart of file transfer

We discuss the following topics in this chapter:

- ▶ 6.1, “Introduction to the business scenario” on page 234
- ▶ 6.2, “Architectural diagram” on page 240
- ▶ 6.3, “Benefits” on page 244
- ▶ 6.4, “Technical implementation” on page 245
- ▶ 6.5, “Business scenario 1” on page 266
- ▶ 6.6, “Business scenario 2” on page 268
- ▶ 6.7, “Business scenario 3” on page 287
- ▶ 6.8, “Business scenario 4” on page 296
- ▶ 6.9, “Business scenario 5” on page 301
- ▶ 6.10, “Summary” on page 315

How to recreate these scenarios: You can download the configuration files and programs that are used in this chapter from the ITSO Redbooks FTP site. You can use these files to recreate these scenarios in your environment. For download instructions, refer to Appendix A, “Additional material” on page 401.

6.1 Introduction to the business scenario

File Transfer Protocol (FTP) has existed since the late 1970s, and it since has been used as the default method for exchanging files over the network. FTP has been widely utilized by businesses for transferring big chunks of data, both on the intranet or over the Internet for internal and external customers. FTP is still offered as part of many business-to-business (B2B) products.

As with the growth of the Internet, various businesses have also observed an exponential increase in the volume of file transfers, either from application-to-application (A2A) internally or externally to the organization boundaries.

For this purpose, FTP meets the functional requirements of business. However, as file sizes and the volume of files transferred increases, in combination with the increase in applications and customers, many organizations need to look for alternatives. Organizations need support for larger file sizes, improved throughput, the management of simultaneous transfers of multiple files to multiple endpoints, better scalability and integration (A2A), and products that integrate better into the current service-oriented architecture (SOA) and cloud environments.

Organizations have also found that the security, auditing, process control, monitoring, and user interface fail to provide adequate administration of growing FTP solutions.

FTP replacement products must manage existing file transfers better and provide these functions:

- ▶ Security (authentication at both ends of the transfer, and authorization for access to the push or pull target)
- ▶ Auditing (non-repudiation of both sender and receiver credentials)
- ▶ Process control (transfer initiation is script-based, schedule-based, file-system-based [move the file when it appears in a folder])
- ▶ Monitoring (restart failed transfers, send notifications of transfer starts and completions)
- ▶ Scheduling
- ▶ Governance and management that the Managed File Transfer (MFT) solution will bring

6.1.1 Limitations with file transfer protocol

FTP, although adjustable and easy to deploy, is still based on the principle of one-to-one and, in certain cases, one-to-many, but not many-to-many. FTP has often been used to send files from a single sender. FTP has traditionally not been used to send one file to several business partners. Small scripts and solutions can overcome this challenge, but writing and maintaining these scripts are unmanageable tasks. A classic example is in B2B scenarios, where a published file can be received by many business partners. However, there is no automated way for the business to communicate and share files without using collaboration techniques.

With the increase in the number of applications and the amount of data that is transferred over FTP, the true, undiscovered problems have started surfacing and have received major attention from industries utilizing raw FTP for their mission-critical processes.

Customers using FTP list the following problems:

- ▶ Limited reliability

FTP-based designs do not generally handle network or other errors well. Large file transfers that are interrupted often cannot be resumed. FTP is not transactional, so sending a set of files in an all-or-nothing way is not done easily.

- ▶ Limited recoverability

Because FTP deals in basic file units, it cannot know easily that a file transfer is incomplete and cannot recover and restart the transfer.

- ▶ Limited security

FTP often requires user IDs and passwords, which are typically sent in clear-text form over the wire.

- ▶ Limited audit control

Typically, an FTP-based transfer cannot be monitored, logged, or audited easily.

FTP in the past has generally been used by existing applications, which were not engineered to meet the demands of today's businesses. Automation, security, administration, compliance, and adaptability were never concerns with FTP in the past. Organizations today strive to keep up with their day-to-day business requirements and meet their service-level agreements (SLAs).

6.1.2 Problem scenario

ITSO Enterprise was introduced as a fictional organization in Chapter 1, "Introduction" on page 1. ITSO Enterprise has been a centennial organization and has multiple stores throughout the country. For demonstration purposes, we limit the number of stores to three.

Store and warehouse location for ITSO Enterprise

ITSO Enterprise stores are located in three cities in the US:

- ▶ Store in Atlanta (Store ID = ITSOATL)
- ▶ Store in Chicago (Store ID = ITSOCHI)
- ▶ Store in New York (Store ID = ITSONEW)

Apart from these stores, ITSO Enterprise also has a warehouse, which is located in a centrally accessible location from all the stores. Figure 6-1 on page 236 maps the stores and the warehouse.



Figure 6-1 ITSO Enterprise warehouse and stores

Inventory application in the stores and the warehouse

At end of each business day, the ITSO Enterprise stores evaluate their sales and current inventory. It is the responsibility of the warehouse to fulfill the requirements of the stores, based on the current inventory and the forecasted sales. To handle the inventory, the ITSO Enterprise stores send their actual inventory counts everyday to the warehouse.

Eventually, the ITSO Enterprise warehouse determines how much inventory to send to the stores and creates new orders for the external suppliers. The ITSO Enterprise stores use native FTP applications for sending files from the stores to the warehouse, as shown in Figure 6-2 on page 237.

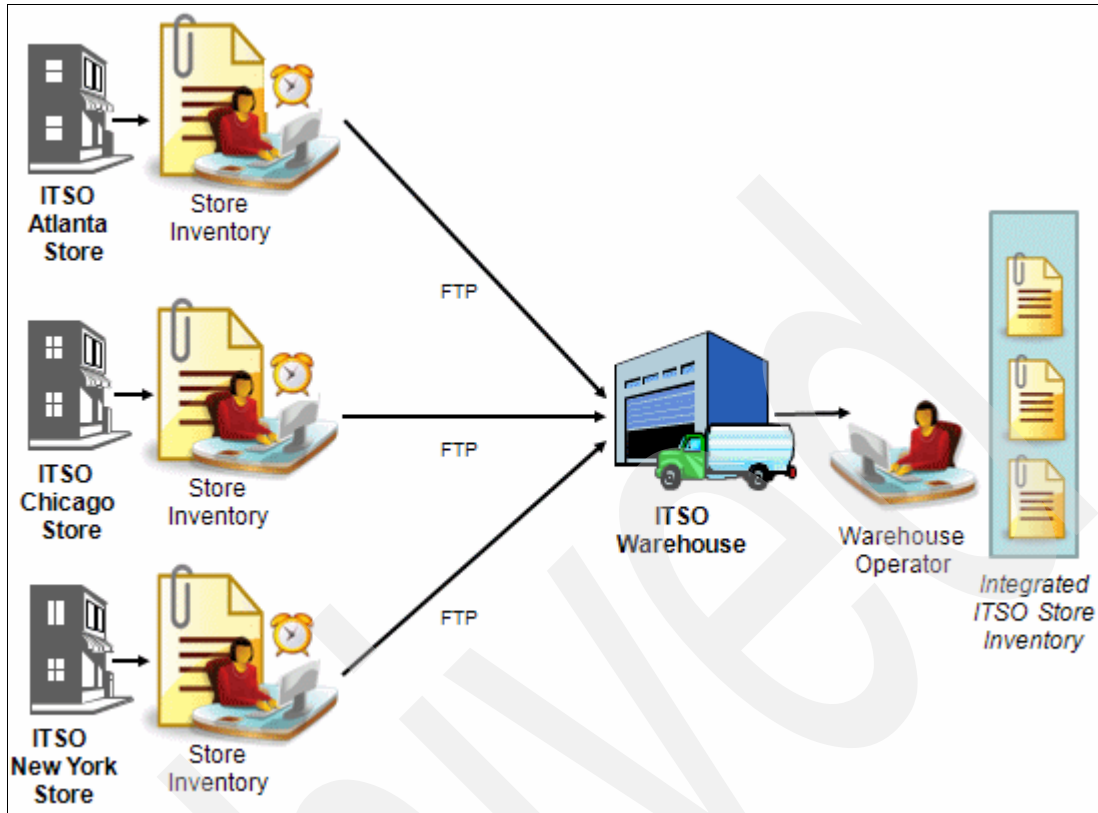


Figure 6-2 ITSO Enterprise traditional file transfer implementation

What happens at the stores

After an ITSO Enterprise store closes for business, the store operator runs a back-end process to evaluate the sales and generates a daily inventory report. The generated inventory reports must be sent before 12:00 a.m. everyday. After the report is generated, the store operator uses an FTP client to send the files to the ITSO Enterprise warehouse FTP server.

What happens at the warehouse

The ITSO Enterprise warehouse receives inventory reports from all of the ITSO Enterprise stores, at a common location in the FTP server. Each day after midnight, the warehouse operator collects all the reports and runs a concatenation program to generate an all-store inventory report. Later, the warehouse operator archives a copy of the same report and also moves the original report to another location on the file system, so that it can be used by another application to make decisions about new orders, fulfillment requirements at the stores, and scheduling the deliveries.

Problems faced by ITSO Enterprise

The ITSO Enterprise stores and warehouse face various problems with the current setup of transferring files:

- ▶ The FTP application in the stores is unable to determine whether the transferred reports were successful or still in progress. There is not a way for ITSO Enterprise to monitor the transfer status.
- ▶ The file transfer logs are not maintained for audit.
- ▶ In the case of a problem during file transfer, the application does not restart the transfer.

- ▶ Not many of the tasks are automated. Many tasks are manual and prone to error.
- ▶ Duplicate file transfers occur, because sometimes a store operator is unable to determine the current status of the transfer, so the store operator sends the file again.
- ▶ Administration is done in silos by various store operators and warehouse operators.
- ▶ Securing the file transfer by using encryption and decryption techniques is not possible or requires an investment in rewriting the code of the FTP clients.

With all these problems in ITSO Enterprise, it is clear that its requirements closely relate to the limitations of FTP, which were discussed in 6.1.1, “Limitations with file transfer protocol” on page 234. This problem was presented to the technical board of ITSO Enterprise. After major discussions, the board decided to implement a solution that is based on emerging technology, such as the Managed File Transfer.

What is Managed File Transfer

A *Managed File Transfer system* introduces control, management, and auditability to address problems that arise when ad hoc file transfers are used to integrate or connect business systems in the enterprise. Figure 6-3 shows several of the advantages of using Managed File Transfer.



Figure 6-3 Managed File Transfer features

6.1.3 Managed File Transfer offering from IBM

IBM WebSphere MQ File Transfer Edition provides an enterprise-grade Managed File Transfer capability that is both robust and easy to use. WebSphere MQ File Transfer Edition exploits the proven reliability and connectivity of WebSphere MQ to transfer files across a wide range of platforms and networks. WebSphere MQ File Transfer Edition takes advantage of existing WebSphere MQ networks, and you can integrate it easily with existing file transfer systems.

Figure 6-4 on page 239 illustrates the concepts of WebSphere MQ File Transfer Edition.

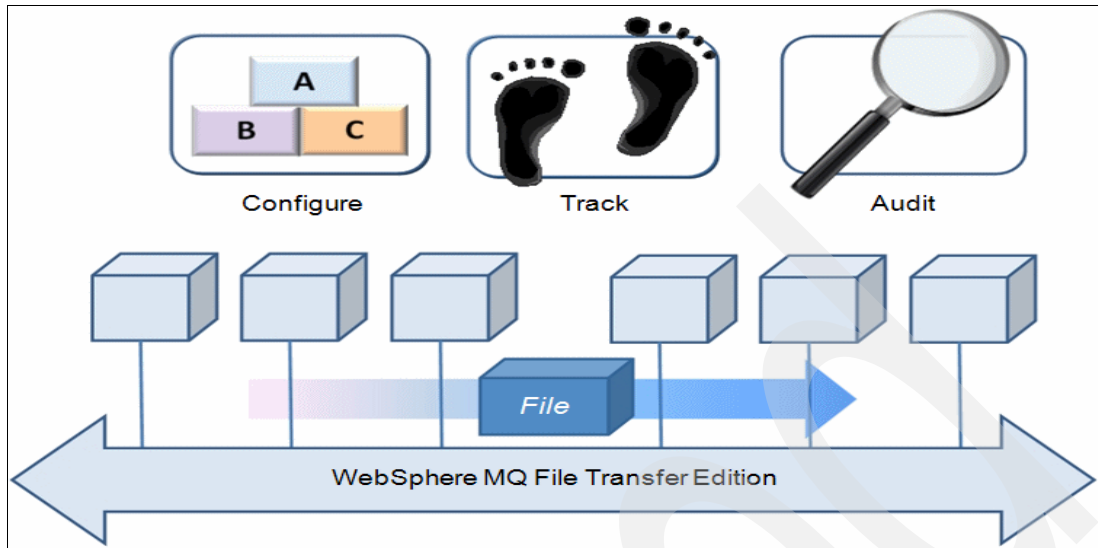


Figure 6-4 Overview of WebSphere MQ File Transfer Edition

WebSphere MQ File Transfer Edition offers the following benefits:

- ▶ Provides reliable Managed File Transfer (MFT) using WebSphere MQ
- ▶ Enables the consolidation of messaging and the file transfer infrastructure onto a single backbone
- ▶ Provides the bulk transfer of files, regardless of their size
- ▶ Offers a record log of transfer activity for audit purposes
- ▶ Offers the automation and control of file movement between IT systems
- ▶ Extends the WebSphere MQ Explorer graphical user interface (GUI) for configuring transfers and monitoring progress remotely
- ▶ Provides scripting support for programmatic control of the transfers
- ▶ Is an SOA-ready approach that enables files to be transferred onto enterprise service buses (ESBs)

WebSphere MQ File Transfer Edition provides enterprise-grade Managed File Transfer using a network of WebSphere MQ queue managers. WebSphere MQ File Transfer Edition is the newest member of the WebSphere MQ family and takes advantage of the proven reliability and wide connectivity choices of WebSphere MQ.

WebSphere MQ File Transfer Edition provides the following features:

- ▶ Delivers robust solutions for Managed File Transfer:
 - Enables control of all aspects of file movement between IT systems
 - Provides file delivery reliability
 - Optimized for both small and massive files
 - Provides audit trail of transfers
- ▶ Designed to integrate with the IBM SOA portfolio
 - Enables files to be delivered to WebSphere Message Broker for file processing
- ▶ Provides secure file transfers using WebSphere MQ security mechanisms

Relationship to WebSphere MQ

WebSphere MQ File Transfer Edition uses WebSphere MQ queue managers to manage file transfers and to carry file data around the network. Figure 6-5 illustrates the relationship between WebSphere MQ File Transfer Edition and WebSphere MQ.

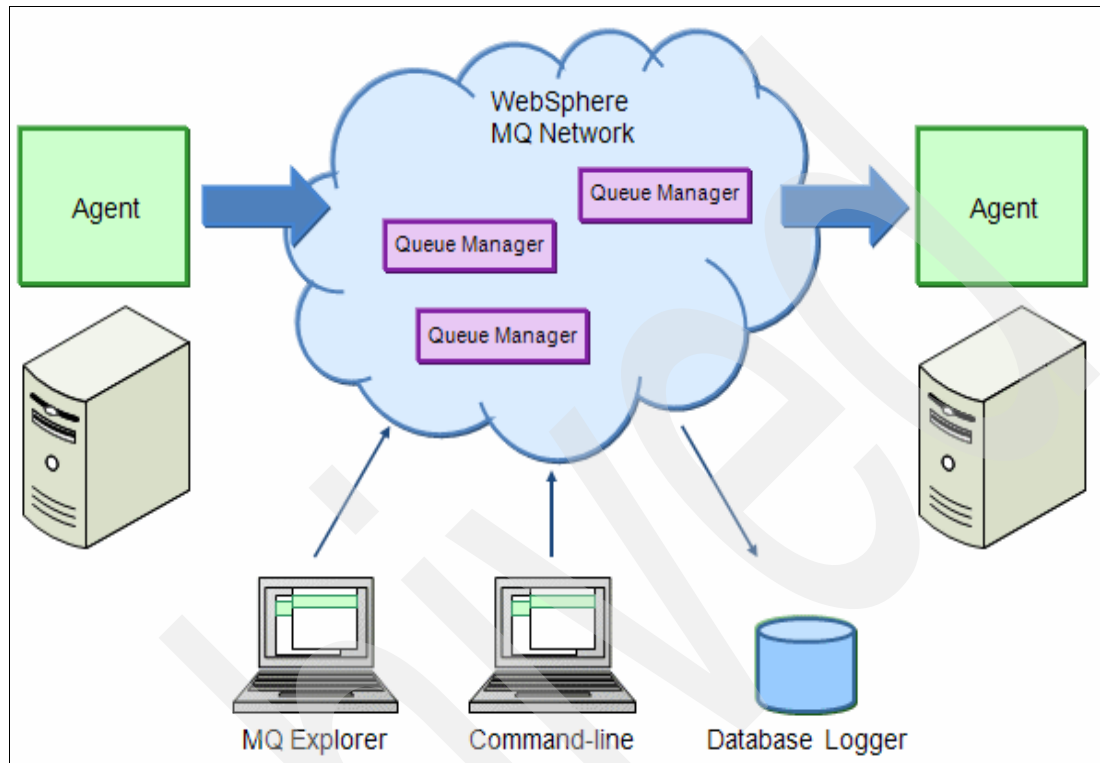


Figure 6-5 Managed File Transfer using WebSphere MQ File Transfer Edition

WebSphere MQ File Transfer Edition uses the following WebSphere MQ queue managers:

- ▶ **Coordination queue manager**
 - ▶ The coordination queue manager receives status and other updates from WebSphere MQ File Transfer Edition agents and publishes the information to interested subscribers.
- ▶ **Agent queue manager**
 - ▶ The WebSphere MQ File Transfer Edition server agents can use their local queue manager or a remote queue manager to send and receive files into the MQ network. WebSphere MQ File Transfer Edition client agents use a remote queue manager to send and receive files.
- ▶ **Command queue manager**
 - ▶ The command-line tools and the WebSphere MQ File Transfer Edition GUI use the command queue manager to communicate with agents and the coordination queue manager.

6.2 Architectural diagram

After the problem was isolated and the solution was identified, responsibility was given to the Solution Architect at ITSO Enterprise to come up with a solution pattern. This architect has already experienced the success stories using Smart SOA approaches and patterns in the

previous scenarios, therefore, the Solution Architect prepares and presents a widely accepted Enterprise Messaging and File Backbone pattern solution to the technical board at ITSO Enterprise.

6.2.1 Enterprise Messaging and File Backbone pattern

Figure 6-6 illustrates the Enterprise Messaging and File Backbone pattern. Implementing a solution using this pattern will help ITSO Enterprise recover from the problems that the company experiences while using native FTP clients and protocol.

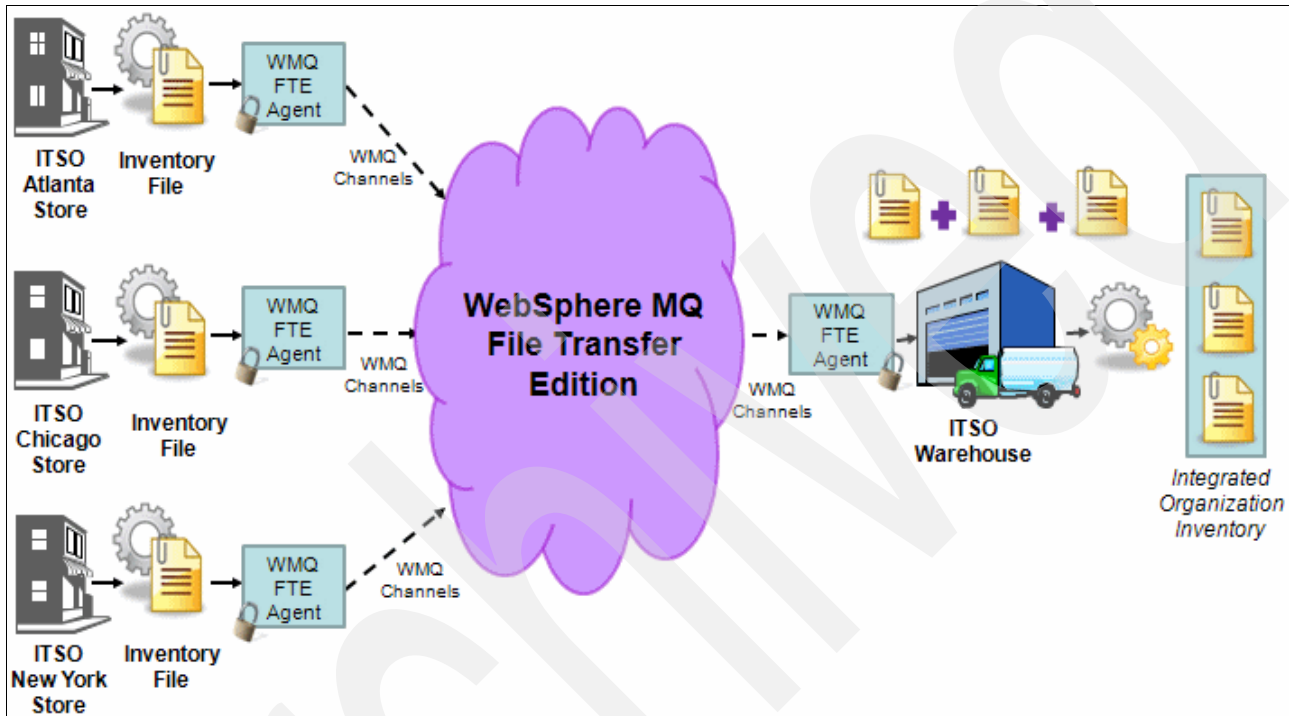


Figure 6-6 Enterprise Messaging and File Backbone pattern for ITSO Enterprise

Figure 6-6 shows a hub and spoke pattern:

- ▶ Hub: WebSphere MQ File Transfer Edition agent on the ITSO Enterprise warehouse acts as a hub
- ▶ Spokes: File Transfer Edition agents on the ITSO Enterprise stores act as spokes

This pattern also opens significant integration opportunities for the ITSO Enterprise stores, which can help extend their store operations with other business units inside the organization. Chapter 7, “Evolve to SOA pattern” on page 317 describes one of the integration opportunities.

Using the Enterprise Messaging and File Backbone pattern with the Managed File Transfer offers the following key features:

- ▶ Transfer of files between systems:
 - Hub and spoke topology
- ▶ Automated and process-driven:
 - Monitoring
 - Triggering

- Scheduling
- Checkpoint and automatic restarts
- Reduction in manual tasks
- ▶ Centralized administration:
 - Reduction of administration silos
 - Auditing and logging
- ▶ Security considerations:
 - Encryption
 - Logging
 - Authentication and authorization
 - Protecting against malicious attacks
 - Secure file system
 - User management

6.2.2 Products utilized to implement this pattern

ITSO Enterprise researched the market-leading software in the Managed File Transfer domain and decided to implement the solution using an IBM stack of products:

- ▶ IBM WebSphere MQ V7.0.1.5
- ▶ IBM WebSphere MQ File Transfer Edition V7.0.4
- ▶ IBM DB2 V9.7

To get more information about these products, refer to these links:

- ▶ IBM WebSphere MQ:
<http://www-01.ibm.com/software/integration/wmq/>
- ▶ IBM WebSphere MQ File Transfer Edition:
<http://www-01.ibm.com/software/integration/wmq/filetransfer/>
- ▶ IBM DB2:
<http://www-01.ibm.com/software/data/db2/>

6.2.3 Addressing ITSO Enterprise issues with WebSphere MQ File Transfer Edition

Using File Transfer Edition over its existing WebSphere MQ infrastructure allows ITSO Enterprise to gain the following key capabilities:

- ▶ Reliability
- ▶ Security
- ▶ Automation
- ▶ Support for any file size
- ▶ Time-independence
- ▶ Consolidated enterprise traffic
- ▶ Integration with non-MQ protocols
- ▶ Centralized control and monitoring

Reliability

Checkpoint-restart, checksum, and MQ assured delivery provide reliable file transfers. These features address the file transfer reliability problems that ITSO Enterprise experienced.

Security

MQ can provide Secure Sockets Layer (SSL) security for all file transfers. All transfers previously using FTP are now secure.

Automation

File transfers do not need to be manual. You can set up scheduled, triggered, and scripted file transfers using File Transfer Edition, which saves ITSO Enterprise the cost of the staff time that was previously required for manual file transfers and eliminates human errors. In addition, File Transfer Edition uses the MQ infrastructure to automatically route files over multiple hops, for example, using multiple machines without having to invoke separate transfers, which is required with FTP.

Support for any file size

File Transfer Edition supports the transfer of files of any size. The checkpoint-restart capability ensures that, in the event of a network or other failure, a large file transfer is resumed where it left off, which allows ITSO Enterprise to distribute large files, such as disk images and software updates, without the slow and time-consuming use of the mail.

Time-independence

ITSO Enterprise can invoke transfers independently of the availability of the network and destination machine. The company can transfer files in an asynchronous manner.

Consolidated enterprise traffic

ITSO Enterprise can now use its existing MQ infrastructure in the stores to support both its enterprise packaged application and file transfer without affecting the performance of its packaged application. ITSO Enterprise can also assign priorities to transfers. In addition, messaging and multiple transfers can use the same port, which means that ITSO Enterprise does not tie up a port with one transfer, which it does with FTP.

Integration with non-MQ protocols

ITSO Enterprise can integrate their file-based infrastructure with other non-MQ protocols through Enterprise Service Bus. Chapter 7, “Evolve to SOA pattern” on page 317 describes one of the integration opportunities.

Centralized control and monitoring

Using the administration utilities of WebSphere MQ File Transfer Edition, ITSO Enterprise can centrally control and monitor the managed file transfer infrastructure.

WebSphere MQ infrastructure: WebSphere MQ infrastructure already existed in the stores, as discussed in Chapter 5, “Message Privacy Enforcement pattern” on page 201.

6.2.4 Systems utilized to implement this scenario

ITSO Enterprise utilized five Microsoft Windows Server 2008 R2 64-bit servers to implement this scenario.

The Domain Name System (DNS) names of these servers are listed:

- ▶ **store.atlanta.itso.com**
The server that is used in the ITSO Enterprise Atlanta store
- ▶ **store.chicago.itso.com**
The server that is used in the ITSO Enterprise Chicago store
- ▶ **store.newyork.itso.com**
The server that is used in the ITSO Enterprise New York store
- ▶ **warehouse.itso.com**
The server that is used in the ITSO Enterprise warehouse
- ▶ **fileadmin.itso.com**
The centralized administration server

Figure 6-7, “Servers that were used by ITSO Enterprise at various locations” on page 244 illustrates the servers that are used to implement the Enterprise Messaging and File Backbone pattern.

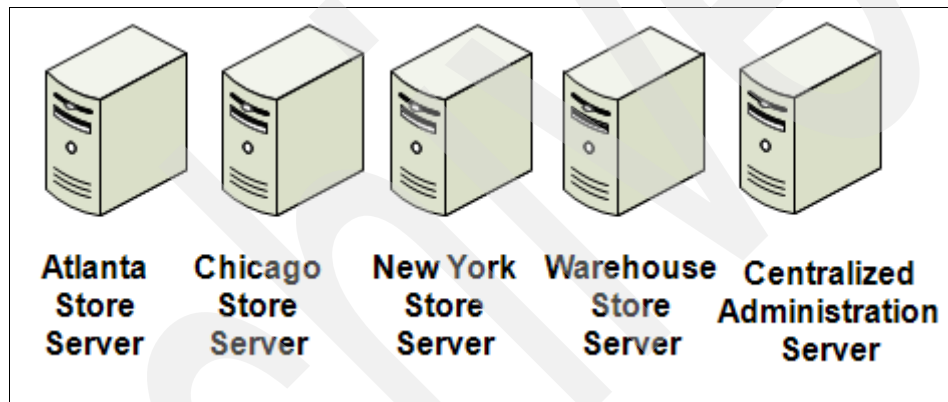


Figure 6-7 Servers that were used by ITSO Enterprise at various locations

6.3 Benefits

There are multiple benefits after ITSO Enterprise implements the Enterprise Messaging and File Backbone pattern. The Solution Architect categorized the benefits into two major categories:

- ▶ Short-term benefits
- ▶ Long-term benefits

6.3.1 Short-term benefits

The Enterprise Messaging and File Backbone pattern offers ITSO Enterprise the following short-term benefits:

- ▶ **Visibility:** Allows ITSO Enterprise to examine the existing file transfer, systems, and the users who send and receive files
- ▶ **Monitoring:** Empowers ITSO Enterprise with monitoring of the system and files through automation

- ▶ **Security:** Integrates easily with various patterns, such as peer-to-peer and hub and spoke, to enable full compliance, identity, access, and authentication
- ▶ **Reporting and auditing:** Compiles and presents data that is related to file transfers, which also allows complete auditing capabilities
- ▶ **Workflow and automation:** Enables ITSO Enterprise to automate the processes that are associated with file transfer and reduce human intervention
- ▶ **Restart and recovery:** Recovers and restarts file transfers in the case of network, power, or another component failure

6.3.2 Long-term benefits

The Enterprise Messaging and File Backbone pattern offers ITSO Enterprise the following long-term benefits:

- ▶ **Adaptability:** Integration opportunities are the key factors which ITSO Enterprise can look forward to long term. WebSphere MQ File Transfer Edition uses the WebSphere MQ network. This feature enables many integration opportunities with other WebSphere MQ-based technologies, such as enterprise messaging, Java Message Service (JMS), and Web 2.0, as illustrated in Figure 6-8.

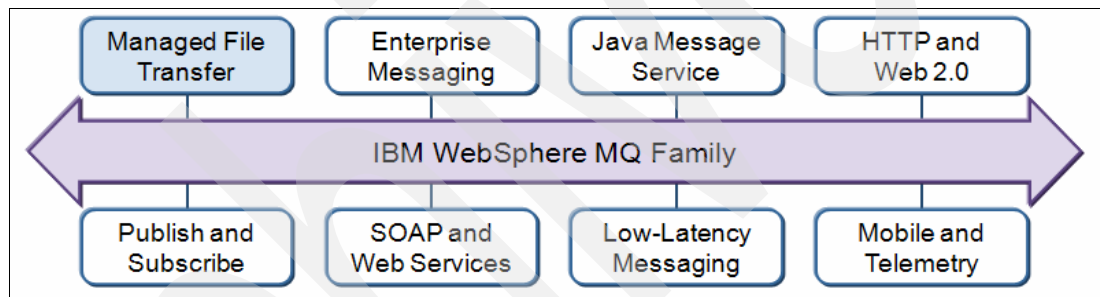


Figure 6-8 Integrating Managed File Transfer as part of WebSphere MQ network

- ▶ **Provisioning:** The Enterprise Messaging and File Backbone pattern enables a rapid path for the new integration opportunities.

WebSphere MQ File Transfer Edition: 6.4.3, “Preparing WebSphere MQ File Transfer Edition” on page 255 describes and demonstrates the integration of WebSphere MQ File Transfer Edition with WebSphere Message Broker.

- ▶ **Cost savings:** The Enterprise Messaging and File Backbone pattern helps ITSO Enterprise achieve process automation and efficiency gains throughout the organization, thereby reducing errors, manual effort, operating expenses, and human-intensive operations.

6.4 Technical implementation

In this section, we discuss the technical implementation of the solution.

6.4.1 Topology

When designing a file transfer topology for WebSphere MQ File Transfer Edition, you can choose to build a topology that supports everything the product has to offer, or you can build a smaller topology that supports only the simple file transfer clients and qualities of service (QoS) that you need. For implementing the multiple site hub and spoke pattern integration, we use the multiple queue manager topology. Table 5-1 lists the components to be created for the Managed File Transfer topology.

Table 6-1 Components for Enterprise Messaging and File Backbone pattern

Server name	Queue manager name	File Transfer Edition agent name
store.atlanta.itso.com	STORE.ATLANTA.QM	STORE.ATLANTA.QM
store.chicago.itso.com	STORE.CHICAGO.QM	STORE.CHICAGO.QM
store.newyork.itso.com	STORE.NEWYORK.QM	STORE.NEWYORK.QM
warehouse.itso.com	WAREHOUSE.AGENT.QM	WAREHOUSE.AGENT.QM
fileadmin.itso.com	ITSO.FILE.COMMAND.QM	
fileadmin.itso.com		

Figure 6-9 reveals the ITSO Enterprise WebSphere MQ topology for the File Transfer Edition implementation.

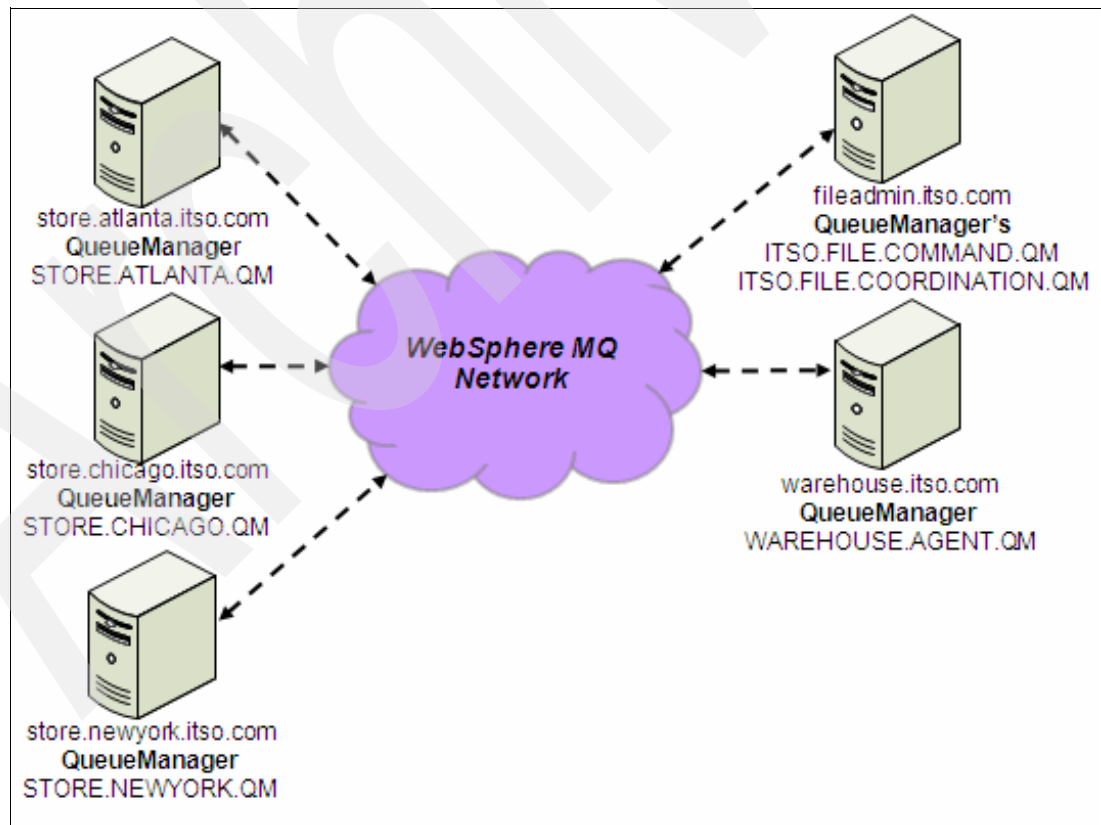


Figure 6-9 WebSphere MQ topology for File Transfer Edition

6.4.2 Preparing WebSphere MQ

WebSphere MQ File Transfer Edition uses WebSphere MQ to transport files between separate systems. To allow WebSphere MQ File Transfer Edition to transfer files between all ITSO Enterprise stores and warehouse systems, you must create the base MQ infrastructure.

We follow these steps before the File Transfer Edition objects are created:

- ▶ Create queue managers
- ▶ Create listener objects on the queue managers
- ▶ Create queue manager clusters
- ▶ Create server connection channels

Creating queue managers

You can create the queue managers by using the `crtmqm` command. Example 6-1 shows the syntax of this command.

Example 6-1 The `crtmqm` command syntax

```
crtmqm [-z] [-q] [-c Text] [-d DefXmitQ] [-h MaxHandles]
      [-md DataPath] [-g ApplicationGroup] [-ss | -sa | -sax | -si]
      [-t TrigInt] [-u DeadQ] [-x MaxUMsgs] [-lp LogPri] [-ls LogSec]
      [-lc | -ll] [-lf LogFileSize] [-ld LogPath] QMgrName
```

To get more information about all of the parameters for the `crtmqm` command, refer to this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/topic/com.ibm.mq.amqzag.doc/fa15650_.htm

Atlanta store queue manager for File Transfer Edition agent

We executed the command that is shown in Example 6-2 on server `store.atlanta.itso.com` to create the queue manager `STORE.ATLANTA.QM`.

Example 6-2 Create queue manager `STORE.ATLANTA.QM`

```
crtmqm STORE.ATLANTA.QM
WebSphere MQ queue manager created.
Directory 'C:\WMQ\qmgrs\STORE!ATLANTA!QM' created.
Creating or replacing default objects for STORE.ATLANTA.QM.
Default objects statistics : 68 created. 0 replaced. 0 failed.
Completing setup.
Setup completed.
```

Chicago store queue manager for File Transfer Edition agent

We executed the command that is shown in Example 6-3 on server `store.chicago.itso.com` to create the queue manager `STORE.CHICAGO.QM`.

Example 6-3 Create queue manager `STORE.CHICAGO.QM`

```
crtmqm STORE.CHICAGO.QM
WebSphere MQ queue manager created.
Directory 'C:\WMQ\qmgrs\STORE!CHICAGO!QM' created.
Creating or replacing default objects for STORE.CHICAGO.QM.
Default objects statistics : 68 created. 0 replaced. 0 failed.
Completing setup.
```

Setup completed.

New York store queue manager for File Transfer Edition agent

We executed the command that is shown in Example 6-4 on server store.newyork.itso.com to create the queue manager STORE.NEWYORK.QM.

Example 6-4 Create queue manager STORE.NEWYORK.QM

crtmqm STORE.NEWYORK.QM

WebSphere MQ queue manager created.
Directory 'C:\WMQ\qmgrs\STORE!NEWYORK!QM' created.
Creating or replacing default objects for STORE.NEWYORK.QM.
Default objects statistics : 68 created. 0 replaced. 0 failed.
Completing setup.
Setup completed.

Warehouse queue manager for File Transfer Edition agent

We executed the command that is shown in Example 6-5 on server warehouse.itso.com to create the queue manager WAREHOUSE.AGENT.QM.

Example 6-5 Create queue manager WAREHOUSE.AGENT.QM

crtmqm WAREHOUSE.AGENT.QM

WebSphere MQ queue manager created.
Directory 'C:\WMQ\qmgrs\WAREHOUSE!AGENT!QM' created.
Creating or replacing default objects for WAREHOUSE.AGENT.QM.
Default objects statistics : 68 created. 0 replaced. 0 failed.
Completing setup.
Setup completed.

Creating the coordination queue manager for File Transfer Edition

We executed the command that is shown in Example 6-6 on server fileadmin.itso.com to create the queue manager ITSO.FILE.COORDINATION.QM.

Example 6-6 Create queue manager ITSO.FILE.COORDINATION.QM

crtmqm ITSO.FILE.COORDINATION.QM

WebSphere MQ queue manager created.
Directory 'C:\WMQ\qmgrs\ITSO!FILE!COORDINATION!QM' created.
Creating or replacing default objects for ITSO.FILE.COORDINATION.QM.
Default objects statistics : 68 created. 0 replaced. 0 failed.
Completing setup.
Setup completed.

Creating the command queue manager for File Transfer Edition

We executed the command that is shown in Example 6-7 on server fileadmin.itso.com to create the queue manager ITSO.FILE.COMMAND.QM.

Example 6-7 Create queue manager ITSO.FILE.COMMAND.QM

crtmqm ITSO.FILE.COMMAND.QM

WebSphere MQ queue manager created.
Directory 'C:\WMQ\qmgrs\ITSO!FILE!COMMAND!QM' created.
Creating or replacing default objects for ITSO.FILE.COMMAND.QM.
Default objects statistics : 68 created. 0 replaced. 0 failed.

Completing setup.
Setup completed.

Now that all the queue managers are created, we start them by using the **strmqm** command. Example 6-8 shows the syntax of this command.

Example 6-8 The syntax of the strmqm command

```
strmqm [-z] [-a | -c | -r | -x] [-d none|minimal|all] [-f]
        [-ns] [-ss | -si] [QMgrName]
```

To get more information about all of the parameters for the **strmqm** command, refer to this website:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/topic/com.ibm.mq.amqzag.doc/fa16090_.htm

Example 6-9 shows the commands to start the various queue managers.

Example 6-9 Start queue manager commands

```
strmqm STORE.ATLANTA.QM
strmqm STORE.CHICAGO.QM
strmqm STORE.NEWYORK.QM
strmqm WAREHOUSE.AGENT.QM
strmqm ITSO.FILE.COORDINATION.QM
strmqm ITSO.FILE.COMMAND.QM
```

Creating listener objects on the queue manager

In this section, we create listener objects on the various queue managers by issuing commands on the WebSphere MQ **runmqsc** console. Also, the listeners' control is set to be the queue manager, so that listeners start and stop based on the control of the queue manager.

To get more information about defining listeners, refer to this website:

http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/topic/com.ibm.mq.csqzaj.doc/sc11050_.htm

Example 6-10 to Example 6-15 on page 250 illustrate the commands to create the listeners on various queue managers.

Example 6-10 Create listener on the queue manager STORE.ATLANTA.QM

```
DEFINE LISTENER ('STORE.LISTENER')
TRPTYPE(TCP)
IPADDR('store.atlanta.itso.com')
PORT(1414) BACKLOG(0)
DESCR('Atlanta Queue Manager Listener')
CONTROL(QMGR) REPLACE
```

Example 6-11 Create listener on the queue manager STORE.CHICAGO.QM

```
DEFINE LISTENER ('STORE.LISTENER')
TRPTYPE(TCP)
IPADDR('store.chicago.itso.com')
PORT(1414) BACKLOG(0)
```

```
DESCR('Chicago Queue Manager Listener')
CONTROL(QMGR) REPLACE
```

Example 6-12 Create listener on the queue manager STORE.NEWYORK.QM

```
DEFINE LISTENER ('STORE.LISTENER')
TRPTYPE(TCP)
IPADDR('store.newyork.itso.com')
PORT(1414) BACKLOG(0)
DESCR('Newyork Queue Manager Listener')
CONTROL(QMGR) REPLACE
```

Example 6-13 Create listener on the queue manager WAREHOUSE.AGENT.QM

```
DEFINE LISTENER ('WAREHOUSE.LISTENER')
TRPTYPE(TCP)
IPADDR('warehouse.itso.com')
PORT(1414) BACKLOG(0)
DESCR('Warehouse Queue Manager Listener')
CONTROL(QMGR) REPLACE
```

Example 6-14 Create listener on the queue manager ITSO.FILE.COORDINATION.QM

```
DEFINE LISTENER ('COORDINATION.LISTENER')
TRPTYPE(TCP)
IPADDR('fileadmin.itso.com')
PORT(1416) BACKLOG(0)
DESCR('FTE Coordination Queue Manager Listener')
CONTROL(QMGR) REPLACE
```

Example 6-15 Create listener on the queue manager ITSO.FILE.COMMAND.QM

```
DEFINE LISTENER ('COMMAND.LISTENER')
TRPTYPE(TCP)
IPADDR('fileadmin.itso.com')
PORT(1415) BACKLOG(0)
DESCR('FTE Command Queue Manager Listener')
CONTROL(QMGR) REPLACE
```

After all the listeners are created, use the command in Example 6-16 to start the listeners from the **runmqsc** console.

Example 6-16 Start queue manager listener objects

```
START LISTENER('listener name')
```

To get more information about this command, refer to this website:

http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/topic/com.ibm.mq.csqzaj.doc/sc13170_.htm

Creating a queue manager cluster

The following channels are required to set up message communication between the file transfer queue managers:

- ▶ Channel from the agent queue manager to the coordination queue manager
- ▶ Channel from the command queue manager to the agent queue manager

- ▶ Channel from the command queue manager to the coordination queue manager
- ▶ Channel from the agent queue manager to any other agent queue manager in the WebSphere MQ File Transfer Edition network

If all of ITSO Enterprise systems are members of an MQ cluster, WebSphere MQ can manage the channels automatically. Using an MQ cluster results in simpler administration and configuration of the MQ network. The benefits of MQ clusters quickly increase as the number of systems grows. Using MQ clusters, the number of channels that must be manually defined is greatly reduced.

For scenarios with larger numbers of queue managers, using MQ clusters offers even greater administrative savings. Therefore, ITSO Enterprise decides to set up an MQ cluster called INVENTORY.CLUSTER. Table 6-2 lists the queue managers that participate in INVENTORY.CLUSTER.

Table 6-2 Participating queue manager in cluster INVENTORY.CLUSTER

Queue manager	Repository
ITSO.FILE.COMMAND.QM	Full repository
ITSO.FILE.COORDINATION.QM	Full repository
STORE.ATLANTA.QM	Partial repository
STORE.CHICAGO.QM	Partial repository
STORE.NEWYORK.QM	Partial repository
WAREHOUSE.AGENT.QM	Partial repository

Figure 6-10 on page 252 illustrates the cluster channels that were created on these queue managers.

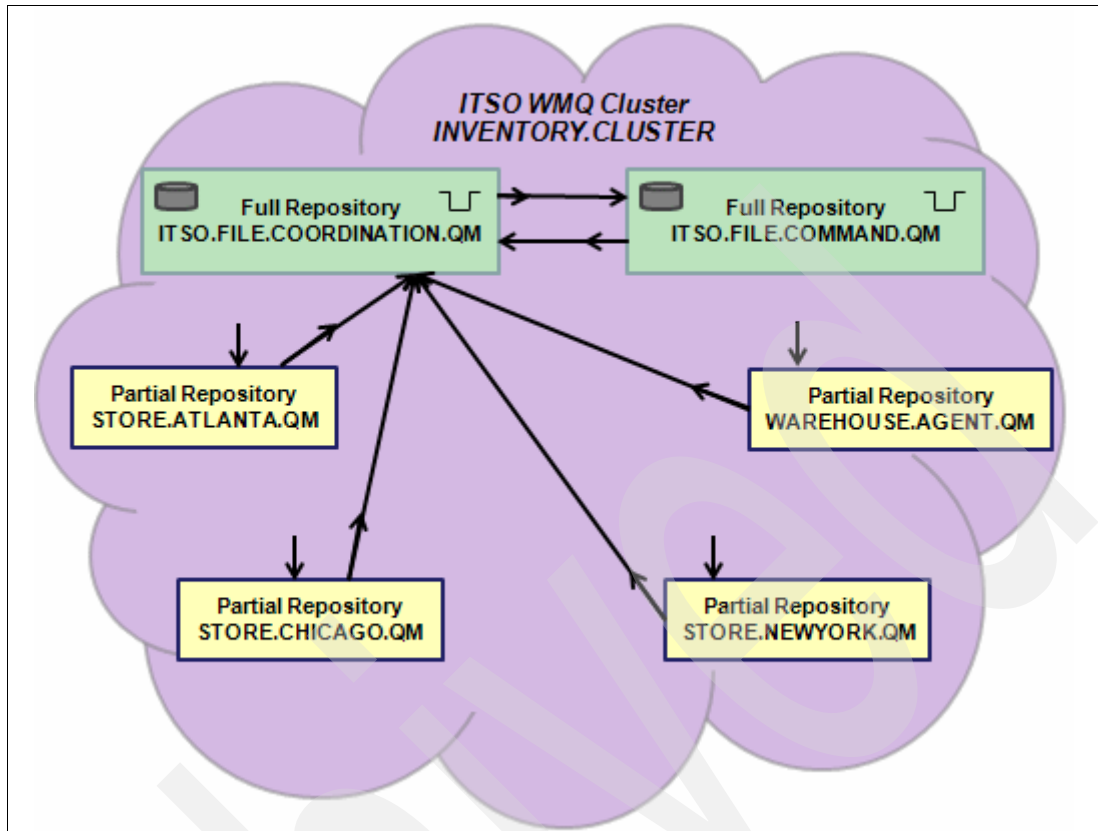


Figure 6-10 Cluster channels in MQ cluster INVENTORY.CLUSTER

Altering the queue manager definitions to add to the repository

On each queue manager that is to hold a full repository, you need to alter the queue manager definition, using the ALTER QMGR command and specifying the REPOS attribute. Issue the command that is shown in Example 6-17 from the `runmqsc` console of the queue managers ITSO.FILE.COMMAND.QM and ITSO.FILE.COORDINATION.QM. These two queue managers will be changed to full repository.

Example 6-17 Change queue manager to full repository

```
ALTER QMGR REPOS('INVENTORY.CLUSTER')
```

Creating the cluster channel on ITSO.FILE.COMMAND.QM

Create the cluster receiver and cluster sender channels on the queue manager ITSO.FILE.COMMAND.QM. Example 6-18 illustrates the commands to create these channels.

Example 6-18 Cluster channels on ITSO.FILE.COMMAND.QM

```
DEFINE CHANNEL (TO.ITSO.FILE.CMD.QM) CHLTYPE (CLUSRCVR) TRPTYPE (TCP) CONNAME
('fileadmin.itso.com(1415)') CLUSTER (INVENTORY.CLUSTER)

DEFINE CHANNEL (TO.ITSO.FILE.CORD.QM) CHLTYPE (CLUSDR) TRPTYPE (TCP) CONNAME
('fileadmin.itso.com(1416)') CLUSTER (INVENTORY.CLUSTER)
```

Creating the cluster channel on ITSO.FILE.COORDINATION.QM

Create the cluster receiver and cluster sender channels on the queue manager ITSO.FILE.COORDINATION.QM. Example 6-19 illustrates the commands to create these channels.

Example 6-19 Cluster channels on ITSO.FILE.COORDINATION.QM

```
DEFINE CHANNEL (TO.ITSO.FILE.CORD.QM) CHLTYPE (CLUSRCVR) TRPTYPE (TCP) CONNAME ('fileadmin.itso.com(1416)') CLUSTER (INVENTORY.CLUSTER)
```

```
DEFINE CHANNEL (TO.ITSO.FILE.CMD.QM) CHLTYPE (CLUSSDR) TRPTYPE (TCP) CONNAME ('fileadmin.itso.com(1415)') CLUSTER (INVENTORY.CLUSTER)
```

Creating the cluster channel on STORE.ATLANTA.QM

Create the cluster receiver and cluster sender channels on the queue manager STORE.ATLANTA.QM. Example 6-20 illustrates the commands to create these channels.

Example 6-20 Cluster channels on STORE.ATLANTA.QM

```
DEFINE CHANNEL (TO.STORE.ATLANTA.QM) CHLTYPE (CLUSRCVR) TRPTYPE (TCP) CONNAME ('store.atlanta.itso.com(1414)') CLUSTER (INVENTORY.CLUSTER)
```

```
DEFINE CHANNEL (TO.ITSO.FILE.CORD.QM) CHLTYPE (CLUSSDR) TRPTYPE (TCP) CONNAME ('fileadmin.itso.com(1416)') CLUSTER (INVENTORY.CLUSTER)
```

Creating the cluster channel on STORE.CHICAGO.QM

Create the cluster receiver and cluster sender channels on the queue manager STORE.CHICAGO.QM. Example 6-21 illustrates the commands to create these channels.

Example 6-21 Cluster channels on STORE.CHICAGO.QM

```
DEFINE CHANNEL (TO.STORE.CHICAGO.QM) CHLTYPE (CLUSRCVR) TRPTYPE (TCP) CONNAME ('store.chicago.itso.com(1414)') CLUSTER (INVENTORY.CLUSTER)
```

```
DEFINE CHANNEL (TO.ITSO.FILE.CORD.QM) CHLTYPE (CLUSSDR) TRPTYPE (TCP) CONNAME ('fileadmin.itso.com(1416)') CLUSTER (INVENTORY.CLUSTER)
```

Creating the cluster channel on STORE.NEWYORK.QM

Create the cluster receiver and cluster sender channels on the queue manager STORE.NEWYORK.QM. Example 6-22 illustrates the commands to create these channels.

Example 6-22 Cluster channels on STORE.NEWYORK.QM

```
DEFINE CHANNEL (TO.STORE.NEWYORK.QM) CHLTYPE (CLUSRCVR) TRPTYPE (TCP) CONNAME ('store.newyork.itso.com(1414)') CLUSTER (INVENTORY.CLUSTER)
```

```
DEFINE CHANNEL (TO.ITSO.FILE.CORD.QM) CHLTYPE (CLUSSDR) TRPTYPE (TCP) CONNAME ('fileadmin.itso.com(1416)') CLUSTER (INVENTORY.CLUSTER)
```

Creating the cluster channel on WAREHOUSE.AGENT.QM

Create the cluster receiver and cluster sender channels on the queue manager WAREHOUSE.AGENT.QM. Example 6-23 on page 254 illustrates the commands to create these channels.

Example 6-23 Cluster channels on WAREHOUSE.AGENT.QM

```
DEFINE CHANNEL (TO.WAREHOUSE.AGNT.QM) CHLTYPE (CLUSRCVR) TRPTYPE (TCP) CONNAME ('warehouse.itso.com(1414)') CLUSTER (INVENTORY.CLUSTER)
```

```
DEFINE CHANNEL (TO.ITSO.FILE.CORD.QM) CHLTYPE (CLUSDR) TRPTYPE (TCP) CONNAME ('fileadmin.itso.com(1416)') CLUSTER (INVENTORY.CLUSTER)
```

The queue managers are added to the repository, and the channels are created between the cluster components. We can also manage this cluster from WebSphere MQ Explorer. Follow the steps:

1. On the server fileadmin.itso.com, select **Start** → **Programs** → **IBM WebSphere MQ** → **WebSphere MQ Explorer**, and click **WebSphere MQ Explorer** to launch the Eclipse GUI.
2. In the Navigator view of WebSphere MQ Explorer, expand **Queue Manager Cluster** → **INVENTORY.CLUSTER**.
3. Figure 6-11 shows the participating queue managers in MQ cluster INVENTORY.CLUSTER.

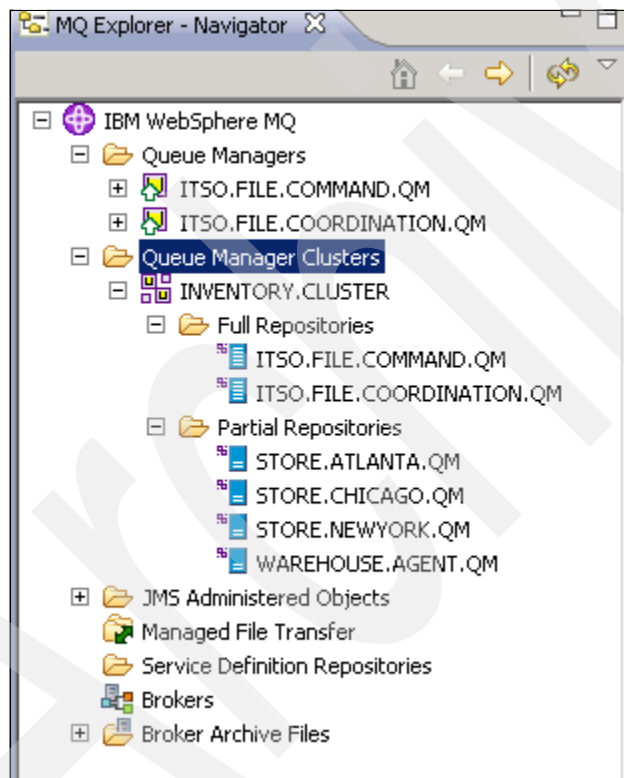


Figure 6-11 Cluster administration from WebSphere MQ Explorer

To get more information about WebSphere MQ clusters, refer to this site:

http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/topic/com.ibm.mq.csqzah.doc/qc10120_.htm

Creating the server connection channel

Message Queue Interface (MQI) channels are used by applications in client mode (rather than bindings mode) to connect to queue managers. MQI channels are bidirectional; they carry WebSphere MQ API calls (for example, "GET a message from queue XYZ") from the

application to the queue manager. They also carry responses to those calls from the queue manager back to the application.

WebSphere MQ File Transfer Edition uses MQI channels to connect agents in client mode to their agent queue managers, and to connect command applications to their command and coordination queue managers. In the default configuration, these connections are made using a SVRCONN channel called SYSTEM.DEF.SVRCONN, which exists by default on all queue managers. Because of these defaults, you do not need to alter any MQI channels for a basic WebSphere MQ File Transfer Edition installation.

However for security purposes, the new server connection channel ITSO.SSL.SVRCONN is created. We will configure security on this channel in a later part of this chapter. Example 6-24 illustrates the command to execute on all the queue managers that are listed in Table 6-2 on page 251.

Example 6-24 Server connection channel

```
DEFINE CHANNEL ('ITSO.SSL.SVRCONN') CHLTYPE(SVRCONN) REPLACE
```

6.4.3 Preparing WebSphere MQ File Transfer Edition

Each File Transfer Edition component requires access to a queue manager. In the simplest case, the users and agents can all connect to the same queue manager. At the other extreme, it is possible to distribute the agents and users across the network and let WebSphere MQ take care of routing the commands and the data traffic among them. When defining topologies, we discussed the agent queue manager, command queue manager, and coordination queue manager. In this section, we configure the WebSphere MQ File Transfer Edition components that are associated with these queue managers:

- ▶ Agent queue manager

Each File Transfer Edition agent is associated with a single queue manager. The agent and its queue manager can be hosted on the same server or the agent can use a WebSphere MQ client channel to communicate with a remote queue manager. When the agent is deployed, the administrator creates a set of queues for that agent to use. Because the queue names include the agent's name, it is possible for one queue manager to host multiple agents. For reasons that we will explain later, this arrangement is quite common.

- ▶ Command queue manager

File Transfer Edition comes with command-line tools and an Eclipse plug-in to let users and instrumentation issue commands to the various agents in the network. In a small network with two agents that connect to a single queue manager, the tooling can be configured to connect directly to that same queue manager. But in a distributed network with many agent queue managers, it is not practical or necessary to have the tooling connect directly to each agent queue manager to issue commands. Instead, the tooling is configured to connect to a single queue manager that has connectivity to all of the agents, relying on WebSphere MQ to route the command messages accordingly. Whichever queue manager the tooling connects to is referred to as the *command queue manager*. There is nothing special about this queue manager other than that it can resolve a route to each agent queue manager. There is no File Transfer Edition code that is required to run and the only required queues are the temporary dynamic queues that are created by the tooling at run time. There is also no requirement that all users connect to the same command queue manager, and in fact the expected deployment groups sets of users by role onto separate command queue managers.

- ▶ Coordination queue manager

The coordination queue manager is the one component that all of the components in the File Transfer Edition network have in common. Any significant event occurring at the agent results in a message that is published by the agent on the SYSTEM.FTE topic that is hosted on the coordination queue manager. Any user or application that needs to know what is happening in the File Transfer Edition network subscribes to these publications. Both the command-line tools and the Eclipse plug-in require a connection to the coordination queue manager to consume publications. Agents have no such requirement, because the publications can be sent from the agent queue manager across the WebSphere MQ network to the coordination queue manager. There is no actual File Transfer Edition code that is required to run on the coordination queue manager. All of the work is accomplished using native WebSphere MQ functionality. File Transfer Edition requires this queue manager to be at WebSphere MQ V7.0 or later because of the pub/sub features that are used, but other than that, there is nothing special about this queue manager.

In this section, we create the File Transfer Edition components. Follow this step-by-step procedure to configure these objects:

1. Set up the coordination properties.
2. Set up the command properties.
3. Create the File Transfer Edition agent on the stores and the warehouse.
4. Start the File Transfer Edition agents.
5. Validate a simple file transfer.

Setting up the coordination properties

The **fteSetupCoordination** command configures a WebSphere MQ File Transfer Edition coordination queue manager. This **fteSetupCoordination** command creates the `wmqfte.properties` file and `coordination.properties` file for WebSphere MQ File Transfer Edition.

It also creates a file that contains the WebSphere MQ definitions for the coordination queue manager. Example 6-25 illustrates the syntax for the **fteSetupCoordination** command.

Example 6-25 The syntax for the fteSetupCoordination command

```
fteSetupCoordination -coordinationQMGr qmgr_name
                    [-coordinationQMGrHost qmgr_host]
                    [-coordinationQMGrPort qmgr_port]
                    [-coordinationQMGrChannel qmgr_channel]
                    [-f] [-default]
```

To get more information about all of the parameters for the **fteSetupCoordination** command, refer to this website:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.doc/setup_coord_cmd.htm

Execute the command that is illustrated in Example 6-26 on the servers that are listed in Table 6-1 on page 246.

Example 6-26 Set up the coordination properties

```
fteSetupCoordination
-coordinationQMGr ITSO.FILE.COORDINATION.QM
-coordinationQMGrHost fileadmin.itso.com
-coordinationQMGrPort 1416
```

```
-coordinationQMgrChannel ITSO.SSL.SVRCONN
```

The command in Example 6-26 on page 256 creates the `wmqfte.properties` and `coordination.properties` files in the `<dataDirectory>` and `<dataDirectory>/ITSO.FILE.COORDINATION.QM` directories respectively. Figure 6-12 and Figure 6-13 illustrate snippets from the files that were created.

dataDirectory: The value of `dataDirectory` is set in the `install.properties` file, which is located in WMQFTE install home.

```
1 wmqfte.properties
#
#Fri Jun 10 05:59:39 EDT 2011
defaultProperties=ITSO.FILE.COORDINATION.QM
```

Figure 6-12 `wmqfte.properties` file

```
1 coordination.properties
#Fri Jun 10 05:59:39 EDT 2011
coordinationQMgr=ITSO.FILE.COORDINATION.QM
coordinationQMgrHost=fileadmin.itso.com
coordinationQMgrChannel=ITSO.SSL.SVRCONN
coordinationQMgrPort=1416
```

Figure 6-13 `coordination.properties` file

The `fteSetupCoordination` command also provides you with the WebSphere MQ Script Commands (MQSC) commands that are listed in Example 6-27. You must run the commands against your coordination queue manager to configure WebSphere MQ File Transfer Edition. The MQSC commands create a queue, topic, and topic string, update a NAMELIST, and set the coordination queue manager's PSMODE attribute to ENABLED.

Example 6-27 *Configuring the coordination queue manager manually*

```
DEFINE TOPIC('SYSTEM.FTE') TOPICSTR('SYSTEM.FTE') REPLACE
ALTER TOPIC('SYSTEM.FTE') NPMGDLV(ALLAVAIL) PMSGDLV(ALLAVAIL)
DEFINE QLOCAL(SYSTEM.FTE) LIKE(SYSTEM.BROKER.DEFAULT.STREAM) REPLACE
ALTER QLOCAL(SYSTEM.FTE) DESCR('Stream for WMQFTE Pub/Sub interface')
DISPLAY NAMELIST(SYSTEM.QPUBSUB.QUEUE.NAMELIST)
ALTER NAMELIST(SYSTEM.QPUBSUB.QUEUE.NAMELIST) + NAMES(SYSTEM.BROKER.DEFAULT.STREAM
+ SYSTEM.BROKER.ADMIN.STREAM,SYSTEM.FTE)
DISPLAY QMGR PSMODE
ALTER QMGR PSMODE(ENABLED)
```

You can achieve the same results by processing the MQSC script that is available in the `<dataDirectory>/ITSO.FILE.COORDINATION.QM` folder. Example 6-28 shows how to process this script on the coordination queue manager.

Example 6-28 *Configuring the coordination queue manager using an mqsc script*

```
runmqsc ITSO.FILE.COORDINATION.QM < ITSO.FILE.COORDINATION.QM.mqsc
```

To get more information about configuring the coordination queue manager, refer to this site:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.doc/configuring_v7_qms.htm

Setting up the command properties

The `fteSetupCommands` command creates the `command.properties` file. This properties file specifies the details of the queue manager that connects to the WebSphere MQ network when you issue commands.

You can use the `fteSetupCommands` command to create a `command.properties` file in the coordination queue manager directory. The command uses the `install.properties` and `wmqfte.properties` files to determine where to locate the `command.properties` file. Example 6-29 illustrates the syntax for the `fteSetupCommands` command.

Example 6-29 The syntax for the `fteSetupCommands` command

```
fteSetupCommands -connectionQMGr qmgr_name [-connectionQMGrHost qmgr_host]
                 [-connectionQMGrPort qmgr_port]
                 [-connectionQMGrChannel qmgr_channel]
                 [-p ConfigurationOptions] [-f]
```

To get more information about all of the parameters for the `fteSetupCommands` command, refer to this website:

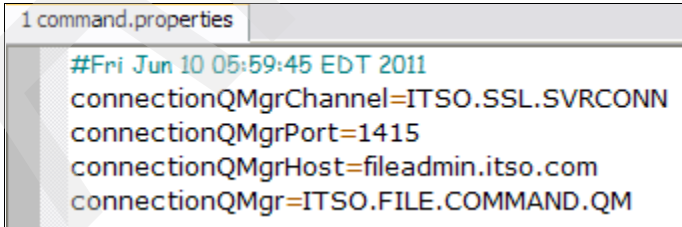
http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.doc/setup_cmds_cmd.htm

Execute the command that is illustrated in Example 6-30 on the servers that are listed in Table 6-1 on page 246.

Example 6-30 `fteSetupCommands` command properties

```
fteSetupCommands
-connectionQMGr ITSO.FILE.COMMAND.QM
-connectionQMGrHost fileadmin.itso.com
-connectionQMGrPort 1415
-connectionQMGrChannel ITSO.SSL.SVRCONN
```

The command in Example 6-30 creates the `command.properties` file in the `<dataDirectory>/ITSO.FILE.COORDINATION.QM` directory. Figure 6-14 illustrates a snippet from the `command.properties` file.



```
1 command.properties
#Fri Jun 10 05:59:45 EDT 2011
connectionQMGrChannel=ITSO.SSL.SVRCONN
connectionQMGrPort=1415
connectionQMGrHost=fileadmin.itso.com
connectionQMGr=ITSO.FILE.COMMAND.QM
```

Figure 6-14 `command.properties` file

Important: Ensure that you have already created and configured a coordination queue manager before you issue the `fteSetupCommands` command.

Creating File Transfer Edition agents on the stores and the warehouse

The **fteCreateAgent** command creates an agent and its associated configuration. Example 6-31 shows the command syntax for the **fteCreateAgent** command.

Example 6-31 The syntax for fteCreateAgent command

```
fteCreateAgent -agentName agent_name -agentQMgr agent_qmgr
               [-agentQMgrHost agent_qmgr_host]
               [-agentQMgrPort agent_qmgr_port]
               [-agentQMgrChannel agent_qmgr_channel]
               [-agentDesc agent_description]
               [-ac] [-p ConfigurationOptions] [-f]
               [-s [service_name] -su service_user
                 [-sp service_password] [-sj service_jvm_options]
                 [-sl log_level]]
               [-n]
```

To get more information about all the parameters for the **fteCreateAgent** command, refer to this website:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.doc/create_agent_cmd.htm

This command provides you with the MQSC commands that you must run against the agent queue manager to create the following agent queues:

- ▶ SYSTEM.FTE.AUTHADM1.agent_name
- ▶ SYSTEM.FTE.AUTHAGT1.agent_name
- ▶ SYSTEM.FTE.AUTHMON1.agent_name
- ▶ SYSTEM.FTE.AUTHOPS1.agent_name
- ▶ SYSTEM.FTE.AUTHSCH1.agent_name
- ▶ SYSTEM.FTE.AUTHTRN1.agent_name
- ▶ SYSTEM.FTE.COMMAND.agent_name
- ▶ SYSTEM.FTE.DATA.agent_name
- ▶ SYSTEM.FTE.EVENT.agent_name
- ▶ SYSTEM.FTE.REPLY.agent_name
- ▶ SYSTEM.FTE.STATE.agent_name

These queues are internal system queues that you must not modify, delete, or read messages from unless you are deleting the agent. The MQSC commands to run are also supplied in a file in the following location:

```
<dataDirectory>\coordination_qmgr\agents\agent_name\agent_name_create.mqsc
```

Each agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent. The `agent.properties` file will be created by the **fteCreateAgent** command. The `agent.properties` file for an agent is located in your `<dataDirectory>/coordination_qmgr_name/agents/agent_name` directory.

Creating a File Transfer Edition agent on the Atlanta store

In Example 6-32, we create a File Transfer Edition agent STORE.ATLANTA.QM, referring to queue manager STORE.ATLANTA.QM. The **fteCreateAgent** command in this example is executed on the store.atlanta.itso.com server.

Example 6-32 fteCreateAgent command on the Atlanta store

```
fteCreateAgent
-agentName STORE.ATLANTA.QM
```

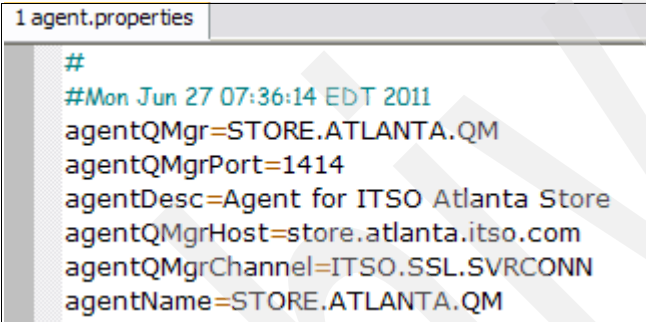
```
-agentQMgr STORE.ATLANTA.QM
-agentQMgrHost store.atlanta.itso.com
-agentQMgrPort 1414
-agentQMgrChannel ITSO.SSL.SVRCONN
-agentDesc "Agent for ITSO Atlanta Store"
```

The command in Example 6-32 on page 259 will also create a script file named `STORE.ATLANTA.QM_create.MQSC`. The `runmqsc` command in Example 6-33 needs to be executed against the queue manager `STORE.ATLANTA.QM` to create the MQ objects for this agent.

Example 6-33 Create MQ objects for Atlanta agent

```
runmqsc STORE.ATLANTA.QM < STORE.ATLANTA.QM_create.MQSC
```

Figure 6-15 illustrates the `agent.properties` file that was created for the File Transfer Edition agent `STORE.ATLANTA.QM`.



```
1 agent.properties
#
#Mon Jun 27 07:36:14 EDT 2011
agentQMgr=STORE.ATLANTA.QM
agentQMgrPort=1414
agentDesc=Agent for ITSO Atlanta Store
agentQMgrHost=store.atlanta.itso.com
agentQMgrChannel=ITSO.SSL.SVRCONN
agentName=STORE.ATLANTA.QM
```

Figure 6-15 *agent.properties* for Atlanta store File Transfer Edition agent

Creating a File Transfer Edition agent on the Chicago store

In Example 6-34, we create a File Transfer Edition agent `STORE.CHICAGO.QM`, referring to queue manager `STORE.CHICAGO.QM`. The `fteCreateAgent` command in this example is executed on the `store.chicago.itso.com` server.

Example 6-34 fteCreateAgent on the Chicago store

```
fteCreateAgent
-agentName STORE.CHICAGO.QM
-agentQMgr STORE.CHICAGO.QM
-agentQMgrHost store.chicago.itso.com
-agentQMgrPort 1414
-agentQMgrChannel ITSO.SSL.SVRCONN
-agentDesc "Agent for ITSO Chicago Store"
```

The command in Example 6-34 also creates a script file named `STORE.CHICAGO.QM_create.MQSC`. The `runmqsc` command in Example 6-35 needs to be executed against the queue manager `STORE.CHICAGO.QM` to create MQ objects for this agent.

Example 6-35 Create MQ objects for the Chicago agent

```
runmqsc STORE.CHICAGO.QM < STORE.CHICAGO.QM_create.MQSC
```

Figure 6-16 illustrates the `agent.properties` file that was created for the File Transfer Edition agent `STORE.CHICAGO.QM`.

```
1 agent.properties
#
#Mon Jun 27 07:36:14 EDT 2011
agentQMgr=STORE.CHICAGO.QM
agentQMgrPort=1414
agentDesc=Agent for ITSO Chicago Store
agentQMgrHost=store.chicago.itso.com
agentQMgrChannel=ITSO.SSL.SVRCONN
agentName=STORE.CHICAGO.QM
```

Figure 6-16 `agent.properties` file for Chicago store File Transfer Edition agent

Creating a File Transfer Edition agent on the New York store

In Example 6-36, we create a File Transfer Edition agent `STORE.NEWYORK.QM`, referring to queue manager `STORE.NEWYORK.QM`. The `fteCreateAgent` command in this example is executed on the `store.newyork.itso.com` server.

Example 6-36 `fteCreateAgent` on the `NEWYORK` store

```
fteCreateAgent
-agentName STORE.NEWYORK.QM
-agentQMgr STORE.NEWYORK.QM
-agentQMgrHost store.newyork.itso.com
-agentQMgrPort 1414
-agentQMgrChannel ITSO.SSL.SVRCONN
-agentDesc "Agent for ITSO NewYork Store"
```

The command in Example 6-36 also creates a script file named `STORE.NEWYORK.QM_create.MQSC`. The `runmqsc` command in Example 6-37 needs to be executed against the queue manager `STORE.NEWYORK.QM` to create MQ objects for this agent.

Example 6-37 `create MQ objects for the NEWYORK agent`

```
runmqsc STORE.NEWYORK.QM < STORE.NEWYORK.QM_create.MQSC
```

Figure 6-17 illustrates the `agent.properties` file that was created for the File Transfer Edition agent `STORE.NEWYORK.QM`.

```
1 agent.properties
#
#Mon Jun 27 07:36:14 EDT 2011
agentQMgr=STORE.NEWYORK.QM
agentQMgrPort=1414
agentDesc=Agent for ITSO NewYork Store
agentQMgrHost=store.newyork.itso.com
agentQMgrChannel=ITSO.SSL.SVRCONN
agentName=STORE.NEWYORK.QM
```

Figure 6-17 `agent.properties` for `NEWYORK` store File Transfer Edition agent

Creating a File Transfer Edition agent on the warehouse

In Example 6-38, we create a File Transfer Edition agent WAREHOUSE.AGENT.QM, referring to queue manager WAREHOUSE.AGENT.QM. The **fteCreateAgent** command in this example is executed on the warehouse.itso.com server.

Example 6-38 fteCreateAgent on the warehouse

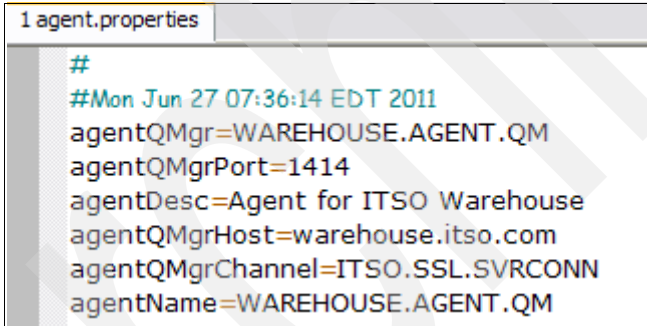
```
fteCreateAgent
-agentName WAREHOUSE.AGENT.QM
-agentQMgr WAREHOUSE.AGENT.QM
-agentQMgrHost warehouse.itso.com
-agentQMgrPort 1414
-agentQMgrChannel ITSO.SSL.SVRCONN
-agentDesc "Agent for ITSO Warehouse"
```

The command in Example 6-38 also creates a script file named WAREHOUSE.AGENT.QM_create.MQSC. The **runmqsc** command in Example 6-39 needs to be executed against the queue manager WAREHOUSE.AGENT.QM to create MQ objects for this agent.

Example 6-39 Create MQ objects

```
runmqsc WAREHOUSE.AGENT.QM < WAREHOUSE.AGENT.QM_create.MQSC
```

Figure 6-18 illustrates the `agent.properties` file that was created for the File Transfer Edition agent WAREHOUSE.AGENT.QM.



```
1 agent.properties
#
#Mon Jun 27 07:36:14 EDT 2011
agentQMgr=WAREHOUSE.AGENT.QM
agentQMgrPort=1414
agentDesc=Agent for ITSO Warehouse
agentQMgrHost=warehouse.itso.com
agentQMgrChannel=ITSO.SSL.SVRCONN
agentName=WAREHOUSE.AGENT.QM
```

Figure 6-18 agent.properties file for warehouse File Transfer Edition agent

Starting the File Transfer Edition agents

You must start an agent before you can use it to perform file transfers. Use the **fteStartAgent** command to start a WebSphere MQ File Transfer Edition agent. The **fteStartAgent** command starts an agent on the system where you issue the command. Because this command is issued locally, you cannot start an agent on a remote system.

The command returns an error if the agent does not start or is already started. The agent communicates with its queue manager based on the values defined in the `agent.properties` file. Example 6-40 shows the syntax for the **fteStartAgent** command.

Example 6-40 The syntax for the fteStartAgent command

```
fteStartAgent [-F] [-p ConfigurationOptions] AgentName
```

To get more information about all of the parameters for the **fteStartAgent** command, refer to this website:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.admin.doc/start_agent_cmd.htm

Example 6-40 on page 262 illustrates the commands that are used to start the File Transfer Edition agents on the ITSO Enterprise stores and the warehouse.

Example 6-41 Start agents on the stores and the warehouse

```
fteStartAgent STORE.ATLANTA.QM
fteStartAgent STORE.CHICAGO.QM
fteStartAgent STORE.NEWYORK.QM
fteStartAgent WAREHOUSE.AGENT.QM
```

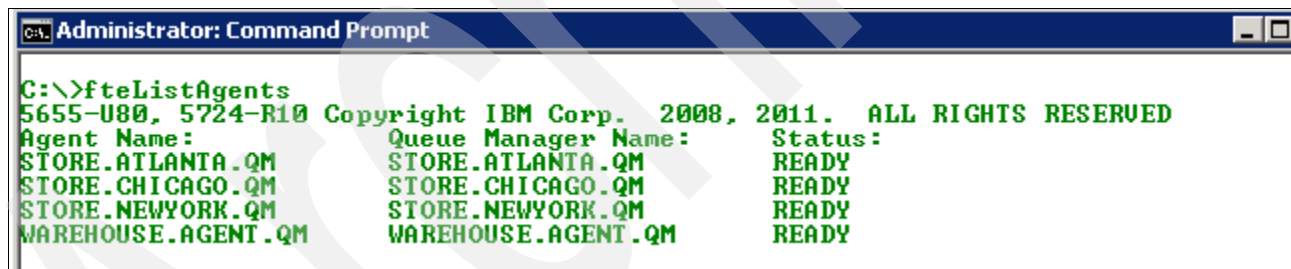
You can list the agents that are registered with a particular queue manager using the command line or the WebSphere MQ Explorer. You can use the **fteListAgents** command, which is shown in Example 6-42, to list all of the WebSphere MQ File Transfer Edition agents that are registered with a particular coordination queue manager from the command line. The following details for each agent are directed to the standard output device:

- ▶ Agent name
- ▶ Agent queue manager
- ▶ Status

Example 6-42 The syntax for fteListAgents command

```
fteListAgents [-p ConfigurationOptions] [-v] [Pattern]
```

Figure 6-19 illustrates the output of the **fteListAgents** command, which is executed on the fileadmin.itso.com server. The output of this command shows that all the agents are ready for file transfer.



```
Administrator: Command Prompt
C:\>fteListAgents
5655-U80, 5724-R10 Copyright IBM Corp. 2008, 2011. ALL RIGHTS RESERVED
Agent Name:          Queue Manager Name:      Status:
STORE.ATLANTA.QM    STORE.ATLANTA.QM        READY
STORE.CHICAGO.QM    STORE.CHICAGO.QM        READY
STORE.NEWYORK.QM    STORE.NEWYORK.QM        READY
WAREHOUSE.AGENT.QM WAREHOUSE.AGENT.QM     READY
```

Figure 6-19 *fteListAgents* output for ITSO Enterprise file agents

You can also get similar details from WebSphere MQ Explorer. Follow these steps to find the status of the registered agents:

1. On the server fileadmin.itso.com, select **Start** → **Programs** → **IBM WebSphere MQ** → **WebSphere MQ Explorer**, and click **WebSphere MQ Explorer** to launch the Eclipse GUI.
2. In the Navigator view of WebSphere MQ Explorer, expand **Managed File Transfer** → **ITSO.FILE.COORDINATION.QM** → **Agents**.

Figure 6-20 on page 264 illustrates the MQ Explorer - Content view with the Agent status and details.

Name	Description	Status	Source transfer	Destination transfer
STORE.ATLANTA.QM	Agent for ITSO Atlanta Store	Ready	0	0
STORE.CHICAGO.QM	Agent for ITSO Chicago Store	Ready	0	0
STORE.NEWYORK.QM	Agent for ITSO NewYork Store	Ready	0	0
WAREHOUSE.AGENT.QM	Agent for ITSO Warehouse	Ready	0	0

Figure 6-20 Agent details in WebSphere MQ Explorer

You can use the **ftePingAgent** command to verify if the agent is Active. Example 6-43 illustrates the syntax for the **ftePingAgent** command.

Example 6-43 The syntax for ftePingAgent command

```
ftePingAgent [-p ConfigurationOptions] [-m QueueManager]
             [-w [timeout]] agentName
```

Validating a simple file transfer

The basic file transfer infrastructure between the agents is configured. We will validate the file transfer between the stores and the warehouse, by manually creating a transfer request. We use the **fteCreateTransfer** command.

The **fteCreateTransfer** command creates and starts a file transfer. Using this command, you can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

You can run the **fteCreateTransfer** command from any system that can connect to the WebSphere MQ network and subsequently route to the agent's queue manager.

This command uses the `command.properties` file to connect to the WebSphere MQ network. If the `command.properties` file does not contain property information, a bindings mode connection is made to the default queue manager on the local system. If the `command.properties` file does not exist, an error is generated. Example 6-44 illustrates the syntax for the command **fteCreateTransfer**.

Example 6-44 The syntax for command fteCreateTransfer

```
fteCreateTransfer [-p ConfigurationOptions] [-w]
                 -sa AgentName [-sm QueueManager]
                 -da AgentName [-dm QueueManager]
                 [-td TransferDefinitionFile]
                 [-gt XMLOutputFile]
                 ([-ss startSchedule] [-tb timeBase]
                 [-oi occurrenceInterval] [-of occurrenceFrequency]
                 [-oc occurrenceCount] | [-es endSchedule])
                 ([-tr triggerSpec ] [-t1])
                 [-jn jobName] [-md metaData] [-cs ChecksumMethod]
                 [-pr <priority>]
                 [-sce <encoding>] [-dce <encoding>] [-dle <line-ending>]
                 [-qmp <true/false>] [-qi]
                 ([-df File] | [-dd Directory] |
                 [-ds SequentialDataset] | [-dp PartitionedDataset] |
```

```
[-dq Queue[@QueueManager]] | [-du fileSpaceName]
([-qs <size>] | [-dqdb <delimiter>] | [-dqdt <pattern>])
[-dqdp <prefix/postfix>]
[-de <action>] [-t TransferMode] [-dqp <true/false>]
[-sd <disposition>] [-r]
[-sq] [-sqgi] [-sqwt]
([-sqdb <delimiter>] | [-sqdt <string>])
[-sqdp <prefix/postfix>]
SourceFileSpec...
```

To get more information about all of the parameters for the **ftCreateTransfer** command, refer to this website:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.doc/start_new_transfer_cmd.htm

The commands in Example 6-45, Example 6-46, and Example 6-47 illustrate the manual file transfer request between the stores and the warehouse.

Example 6-45 Inventory file transfer from the Atlanta store to the warehouse

```
ftcreatetransfer
-sa STORE.ATLANTA.QM
-sm STORE.ATLANTA.QM
-da WAREHOUSE.AGENT.QM
-dm WAREHOUSE.AGENT.QM
-dd "E:\ITSO.INVENTORY.FILES\WAREHOUSE"
"E:\ITSO.INVENTORY.FILES\ATLANTA\ITSOATL.INVENTORY.csv"
```

Example 6-46 Inventory file transfer from the Chicago store to the warehouse

```
ftcreatetransfer
-sa STORE.CHICAGO.QM
-sm STORE.CHICAGO.QM
-da WAREHOUSE.AGENT.QM
-dm WAREHOUSE.AGENT.QM
-dd "E:\ITSO.INVENTORY.FILES\WAREHOUSE"
"E:\ITSO.INVENTORY.FILES\CHICAGO\ITSOCHI.INVENTORY.csv"
```

Example 6-47 Inventory file transfer from the NEWYORK store to the warehouse

```
ftcreatetransfer
-sa STORE.NEWYORK.QM
-sm STORE.NEWYORK.QM
-da WAREHOUSE.AGENT.QM
-dm WAREHOUSE.AGENT.QM
-dd "E:\ITSO.INVENTORY.FILES\WAREHOUSE"
"E:\ITSO.INVENTORY.FILES\NEWYORK\ITSONEW.INVENTORY.csv"
```

From WebSphere MQ Explorer, follow these steps to verify the transfer status:

1. On the server fileadmin.itso.com, select **Start** → **Programs** → **IBM WebSphere MQ** → **WebSphere MQ Explorer**, and click **WebSphere MQ Explorer** to launch the Eclipse GUI.
2. In the Navigator view of WebSphere MQ Explorer, expand **Managed File Transfer** → **ITSO.FILE.COORDINATION.QM** → **Transfer log**.

Figure 6-21 on page 266 shows the MQ Explorer - Content view and the Transfer Log details.

	Source	Destination	Completion State	Owner	Started (America/New_)
+	STORE.CHICAGO.QM	WAREHOUSE.AGENT.QM	Successful	wmquser @ 9.12.5.177	6/27/11 12:40:12 PM EDT
+	STORE.ATLANTA.QM	WAREHOUSE.AGENT.QM	Successful	wmquser @ 9.12.5.177	6/27/11 12:40:14 PM EDT
+	STORE.NEWYORK.QM	WAREHOUSE.AGENT.QM	Successful	wmquser @ 9.12.5.177	6/27/11 12:40:16 PM EDT

Figure 6-21 Transfer log for file transfer validation

6.5 Business scenario 1

In this business scenario, we demonstrate the recovery and restart feature of WebSphere MQ File Transfer Edition. *Checkpoint restart* allows a stalled file transfer to restart from where it got disrupted.

6.5.1 Requirement

ITSO Enterprise stores are enabled with a high-speed satellite network; however occasionally, they face network issues that cause a disruption in file transfers. ITSO Enterprise wants to ensure the inventory report file integrity when the network transmission is broken and then recovered again.

6.5.2 Solution

WebSphere MQ File Transfer Edition can satisfy this requirement. WebSphere MQ File Transfer Edition can identify whether the agent or queue manager is unavailable for any reason, for example, because of a power or network failure. WebSphere MQ File Transfer Edition recovers in the following manner in these scenarios:

- ▶ Typically, if there is a problem while a file is in the process of being transferred, WebSphere MQ File Transfer Edition recovers and restarts that file transfer after the problem is repaired.
- ▶ If a file that was in the process of being transferred is deleted or changed while the agent or queue manager is unavailable, the transfer fails and you get a message in the transfer log that provides the details about the failure.
- ▶ If an agent process fails during a file transfer, the transfer continues when you restart the agent.
- ▶ If an agent loses the connection to its agent queue manager, the agent waits in an infinite loop trying to reconnect to the queue manager. When the agent successfully reconnects to its queue manager, the current transfer continues.
- ▶ If the agent is stopped for any reason, any resource monitors that are associated with an agent stops polling. When the agent recovers, the monitors are also restarted, and resource polling resumes.

Stalled transfers: You can check the status of your transfers in WebSphere MQ Explorer. If any transfers appear as Stalled, you might need to take corrective action, because the Stalled status denotes an issue either with the agent or between the two agents that are involved in the transfer.

6.5.3 Testing checkpoint and restart

In the following steps, ITSO Enterprise begins to test the checkpoint/recovery mechanism of WebSphere MQ File Transfer Edition. In this example, we create a transfer request of a 2 GB inventory file from the Chicago store to the warehouse. Later, we recreate the network problem by disabling the network on warehouse server and, then after a while, enable the network. Follow the steps to simulate this test:

1. Create a file transfer between the Chicago store and the warehouse using the **ftcreatetransfer** command that is shown in Example 6-48.

Example 6-48 File transfer request from the Chicago store to the warehouse

```
ftcreatetransfer
-sa STORE.CHICAGO.QM
-sm STORE.CHICAGO.QM
-da WAREHOUSE.AGENT.QM
-dm WAREHOUSE.AGENT.QM
-cs MD5 -t binary
-df "E:\ITSO.INVENTORY.FILES\WAREHOUSE\CHICAGO.INVENTORY.csv"
"E:\ITSO.INVENTORY.FILES\CHICAGO\CHICAGO.INVENTORY.csv"
```

2. See the Current Transfer Progress view and display the state, as shown in Figure 6-22.

Current File	File Number	Progress	Rate	Started (America/New_York)
CHICAGO.INVENTORY.csv - (760MiB / 2GiB)	1 / 1	<div style="width: 25%; background-color: #0056b3; height: 10px;"></div> 25%	29MiB/s	6/27/11 5:03:56 PM EDT

Figure 6-22 The current transfer state

3. Click **Disable** to disable the local area network on the server warehouse.itso.com (Figure 6-23 on page 268).

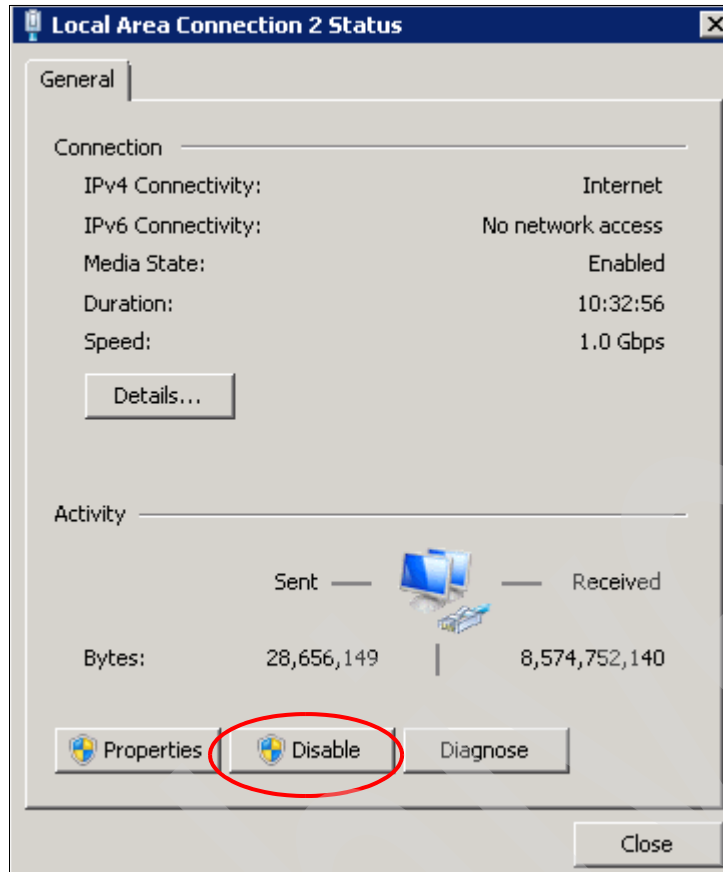


Figure 6-23 Disable the network on the warehouse

4. Enable the local area network on the warehouse after a few minutes.
5. When the WebSphere MQ channels recover between the Chicago store and the warehouse, the file transfer continues, as shown in Figure 6-24.

Administration Log					
Current File	File Number	Progress	Rate	Started (America/New_York)	
CHICAGO.INVENTORY.csv - (982MiB / 2GiB)	1 / 1	<div style="width: 32%; background-color: blue;"></div> 32%	2MiB/s	6/27/11 5:12:17 PM EDT	

Figure 6-24 The current transfer state after the network is enabled

The recovery is complete and successful. ITSO Enterprise is satisfied that this mechanism of WebSphere MQ File Transfer Edition works as intended.

6.6 Business scenario 2

In this section, we describe the file transfer scenario that includes the WebSphere MQ File Transfer Edition resource monitor feature to automate the tasks that were performed manually before. This section discusses a scenario that uses an existing WebSphere MQ network so that WebSphere MQ File Transfer Edition can be used to automate the existing file transfer systems and make them more reliable and auditable.

6.6.1 Requirement

ITSO Enterprise wants the following tasks to be automated:

1. After the inventory files are created on the store servers, they are automatically transferred to the warehouse.
2. If a file transfer fails, an alert is sent to that storeadmin using emails and paging.
3. After all the files are received at the warehouse server, they are concatenated into a single inventory file.
4. The concatenated file is automatically archived and a copy is moved to the inventory processing system.

6.6.2 Solution

WebSphere MQ File Transfer Edition provides tasks that you can use to integrate the file transfer function into the Apache Ant tool.

Ant scripts (or build files) are XML documents defining one or more targets. These targets contain task elements to run. You can use the **fteAnt** command to run Ant tasks in a WebSphere MQ File Transfer Edition environment that you have already configured. You can use file transfer Ant tasks from your Ant scripts to coordinate complex file transfer operations from an interpreted scripting language.

For more information about Apache Ant, see the Apache Ant project web page:

<http://ant.apache.org/>

WebSphere MQ File Transfer Edition provides a number of Ant tasks that you can use to access file transfer capabilities:

- ▶ `fte:awaitoutcome`
- ▶ `fte:call`
- ▶ `fte:cancel`
- ▶ `fte:filecopy`
- ▶ `fte:filemove`
- ▶ `fte:ignoreoutcome`
- ▶ `fte:ping`
- ▶ `fte:uuid`

To get more information about these tasks, refer to this website:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.doc/ant_tasks.htm

We create two Ant scripts:

- ▶ `transfer.xml`: This Ant script moves files from the stores to the warehouse. In case this transfer fails, this script generates email alerts.
- ▶ `concatenate.xml`: This Ant script concatenates and archives the daily inventory file.

WebSphere MQ File Transfer Edition also provides an **fteAnt** command to execute the ant scripts. Developers can use this command to test the functionality of these scripts. Example 6-49 illustrates the syntax for this command.

Example 6-49 The syntax for the fteAnt command

```
fteAnt ([-quiet] | [-verbose]) [(-debug | -d)]
```

```
[(-buildfile | -file | -f) filename]
[[-D property=value [-D property=value] ...]
[(-keep-going | -k)]
[-propertyfile filename] [target [target2 [target3] ...]]
```

transfer.xml Ant script

ITSO Enterprise developed this Ant script to transfer inventory and trigger files from a store to the warehouse. Example 6-50 illustrates the various Ant tasks put together in the script. This example shows the transfer configurations for the Atlanta store.

This Ant script consist of the following sections:

- ▶ Initialize the operational arguments
- ▶ Test the availability of the File Transfer Edition agents
- ▶ Create the transfer request for the inventory and trigger file
- ▶ Assess the transfer and if it fails, send an email to the store administrator

This Ant script uses two files for the Atlanta store:

- ▶ Inventory file: <storeID>.INVENTORY.csv
- ▶ Trigger file: <storeID>.INVENTORY.go

We discuss the trigger file, while creating resource monitors.

Example 6-50 Ant script transfer.xml for the ITSO Enterprise Atlanta store

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns:fte="antlib:com.ibm.wmqfte.ant.taskdefs" default="complete">

  <target name="init" description="initialise task properties">
    <propertyname="cmd.qmgr" value="ITSO.FILE.COMMAND.QM@filadmin.itso.com@1415@ITSO.SSL.SVRCONN"/>
    <property name="src" value="STORE.ATLANTA.QM@STORE.ATLANTA.QM"/>
    <property name="dst" value="WAREHOUSE.AGENT.QM@WAREHOUSE.AGENT.QM"/>
    <property name="mail.host" value="smtp.itso.com"/>
    <property name="mail.to" value="atlantaadmin@itso.com"/>
    <property name="src.file1" value="E:\ITSO.INVENTORY.FILES\ATLANTA\ITSOATL.INVENTORY.csv"/>
    <property name="src.file2" value="E:\ITSO.INVENTORY.FILES\ATLANTA\ITSOATL.INVENTORY.go"/>
    <property name="dst.file1" value="E:\ITSO.INVENTORY.FILES\WAREHOUSE\ITSOATL.INVENTORY.csv"/>
    <property name="dst.file2" value="E:\ITSO.INVENTORY.FILES\WAREHOUSE\ITSOATL.INVENTORY.go"/>
    <fte:uuid property="job.name" length="8" prefix="Transfer Daily Inventory"/>
  </target>

  <target name="ping" depends="init" description="check agents">
    <fte:ping cmdqm="{cmd.qmgr}" agent="{src}" rcproperty="ping.src.rc" timeout="10"/>
    <fte:ping cmdqm="{cmd.qmgr}" agent="{dst}" rcproperty="ping.dst.rc" timeout="10"/>
    <condition property="run.ping">
      <and>
        <equals arg1="{ping.src.rc}" arg2="0"/>
        <equals arg1="{ping.dst.rc}" arg2="0"/>
      </and>
    </condition>
    <fail unless="run.ping" message="*** AGENTS NOT RUNNING ***"/>
  </target>

  <target name="fileTransfer" depends="ping" description="transfer file" if="run.ping">
    <fte:filecopy cmdqm="{cmd.qmgr}" src="{src}" dst="{dst}" jobname="{job.name}"
    rcproperty="copy1.rc">
```

```

        <fte:filespec srcfilespec="{src.file1}" dstfile="{dst.file1}" overwrite="true"
checksum="MD5" conversion="binary"/>
    </fte:filecopy>
    <fte:filecopy cmdqmq="{cmd.qmgr}" src="{src}" dst="{dst}" jobname="{job.name}"
rcproperty="copy2.rc">
        <fte:filespec srcfilespec="{src.file2}" dstfile="{dst.file2}" overwrite="true"
checksum="MD5" conversion="binary"/>
    </fte:filecopy>
    <condition property="file1.transfer.failed">
        <not><equals arg1="{copy1.rc}" arg2="0"/></not>
    </condition>
    <condition property="file2.transfer.failed">
        <not><equals arg1="{copy2.rc}" arg2="0"/></not>
    </condition>
</target>

<target name="email1" depends="fileTransfer" description="send email"
if="file1.transfer.failed">
    <mail mailhost="{mail.host}" mailport="20001" encoding="plain" subject="{job.name}
failed!">
        <from address="fteadmin@itso.com"/>
        <to address="{mail.to}"/>
        <message>Transfer of Inventory file FAILED: {job.name} from agent {src} to agent
{dst} has failed with return code: {copy1.rc}</message>
    </mail>
</target>

<target name="email2" depends="fileTransfer" description="send email"
if="file2.transfer.failed">
    <mail mailhost="{mail.host}" mailport="20001" encoding="plain" subject="{job.name}
failed!">
        <from address="fteadmin@itso.com"/>
        <to address="{mail.to}"/>
        <message>Transfer of trigger file FAILED: {job.name} from agent {src} to agent {dst}
has failed with return code: {copy2.rc}</message>
    </mail>
</target>

<target name="complete" depends="email1, email2"/>
</project>

```

Save this script at the following location, E\INVENTORY\ATLANTA\transfer.xml, on the server named store.atlanta.itso.com. To test this Ant script, use the **fteAnt** command, as shown in Example 6-51.

Example 6-51 Using the fteant command to test the transfer.xml Ant script for the Atlanta store

```
fteant -f transfer.xml
```

After this command executes, you can monitor the transferred file from the WebSphere MQ Explorer Transfer Log view. Figure 6-25 on page 272 illustrates the transfer status.

	Source	Destination	Completion State	Owner	Started (America/New_Yo
+	STORE.ATLANTA.QM	WAREHOUSE.AGENT.QM	Successful	wmquser @ 9.12.5.177	6/28/11 10:27:27 AM EDT
+	STORE.ATLANTA.QM	WAREHOUSE.AGENT.QM	Successful	wmquser @ 9.12.5.177	6/28/11 10:27:27 AM EDT

Figure 6-25 Test results for the transfer.xml ant script

Example 6-52 and Example 6-53 on page 273 illustrate the Ant script transfer.xml for the Chicago and NEWYORK stores.

Example 6-52 Ant script transfer.xml for the Chicago store

```

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns:fte="antlib:com.ibm.wmqfte.ant.taskdefs" default="complete">

<target name="init" description="initialise task properties">
<propertyname="cmd.qmgr" value="ITSO.FILE.COMMAND.QM@filadmin.itso.com@1415@ITSO.SSL.SVRCONN"/>
<property name="src" value="STORE.CHICAGO.QM@STORE.CHICAGO.QM"/>
<property name="dst" value="WAREHOUSE.AGENT.QM@WAREHOUSE.AGENT.QM"/>
<property name="mail.host" value="smtp.itso.com"/>
<property name="mail.to" value="chicagoadmin@itso.com"/>
<property name="src.file1" value="E:\ITSO.INVENTORY.FILES\CHICAGO\ITSOCHI.INVENTORY.csv"/>
<property name="src.file2" value="E:\ITSO.INVENTORY.FILES\CHICAGO\ITSOCHI.INVENTORY.go"/>
<property name="dst.file1" value="E:\ITSO.INVENTORY.FILES\WAREHOUSE\ITSOCHI.INVENTORY.csv"/>
<property name="dst.file2" value="E:\ITSO.INVENTORY.FILES\WAREHOUSE\ITSOCHI.INVENTORY.go"/>
<fte:uuid property="job.name" length="8" prefix="Transfer Daily Inventory" />
</target>

<target name="ping" depends="init" description="check agents">
<fte:ping cmdqm="${cmd.qmgr}" agent="${src}" rcproperty="ping.src.rc" timeout="10"/>
<fte:ping cmdqm="${cmd.qmgr}" agent="${dst}" rcproperty="ping.dst.rc" timeout="10"/>
<condition property="run.ping">
<and>
<equals arg1="${ping.src.rc}" arg2="0"/>
<equals arg1="${ping.dst.rc}" arg2="0"/>
</and>
</condition>
<fail unless="run.ping" message="*** AGENTS NOT RUNNING ***"/>
</target>

<target name="fileTransfer" depends="ping" description="transfer file" if="run.ping">
<fte:filecopy cmdqm="${cmd.qmgr}" src="${src}" dst="${dst}" jobname="${job.name}"
rcproperty="copy1.rc">
<fte:filespec srcfilespec="${src.file1}" dstfile="${dst.file1}" overwrite="true"
checksum="MD5" conversion="binary"/>
</fte:filecopy>
<fte:filecopy cmdqm="${cmd.qmgr}" src="${src}" dst="${dst}" jobname="${job.name}"
rcproperty="copy2.rc">
<fte:filespec srcfilespec="${src.file2}" dstfile="${dst.file2}" overwrite="true"
checksum="MD5" conversion="binary"/>
</fte:filecopy>
<condition property="file1.transfer.failed">

```

```

        <not><equals arg1="{copy1.rc}" arg2="0"/></not>
    </condition>
    <condition property="file2.transfer.failed">
        <not><equals arg1="{copy2.rc}" arg2="0"/></not>
    </condition>
</target>

<target name="email1" depends="fileTransfer" description="send email"
if="file1.transfer.failed">
    <mail mailhost="{mail.host}" mailport="20001" encoding="plain" subject="{job.name}
failed!">
        <from address="fteadmin@itso.com"/>
        <to address="{mail.to}"/>
        <message>Transfer of Inventory file FAILED: {job.name} from agent {src} to agent
{dst} has failed with return code: {copy1.rc}</message>
    </mail>
</target>

<target name="email2" depends="fileTransfer" description="send email"
if="file2.transfer.failed">
    <mail mailhost="{mail.host}" mailport="20001" encoding="plain" subject="{job.name}
failed!">
        <from address="fteadmin@itso.com"/>
        <to address="{mail.to}"/>
        <message>Transfer of trigger file FAILED: {job.name} from agent {src} to agent {dst}
has failed with return code: {copy2.rc}</message>
    </mail>
</target>

<target name="complete" depends="email1, email2"/>
</project>

```

Save this script at the following location, E:\INVENTORY\CHICAGO\transfer.xml, on the server store.chicago.itso.com.

Example 6-53 Ant script Transfer.xml for the NEWYORK store

```

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns:fte="antlib:com.ibm.wmqfte.ant.taskdefs" default="complete">

<target name="init" description="initialise task properties">
<propertyname="cmd.qmgr" value="ITSO.FILE.COMMAND.QM@filadmin.itso.com@1415@ITSO.SSL.SVRCONN"/>
<property name="src" value="STORE.NEWYORK.QM@STORE.NEWYORK.QM"/>
<property name="dst" value="WAREHOUSE.AGENT.QM@WAREHOUSE.AGENT.QM"/>
<property name="mail.host" value="smtp.itso.com"/>
<property name="mail.to" value="newyorkadmin@itso.com"/>
<property name="src.file1" value="E:\ITSO.INVENTORY.FILES\NEWYORK\ITSONEW.INVENTORY.csv"/>
<property name="src.file2" value="E:\ITSO.INVENTORY.FILES\NEWYORK\ITSONEW.INVENTORY.go"/>
<property name="dst.file1" value="E:\ITSO.INVENTORY.FILES\WAREHOUSE\ITSONEW.INVENTORY.csv"/>
<property name="dst.file2" value="E:\ITSO.INVENTORY.FILES\WAREHOUSE\ITSONEW.INVENTORY.go"/>
<fte:uuid property="job.name" length="8" prefix="Transfer Daily Inventory "/>
</target>

<target name="ping" depends="init" description="check agents">
<fte:ping cmdqm="{cmd.qmgr}" agent="{src}" rcproperty="ping.src.rc" timeout="10"/>
<fte:ping cmdqm="{cmd.qmgr}" agent="{dst}" rcproperty="ping.dst.rc" timeout="10"/>

```

```

<condition property="run.ping">
  <and>
    <equals arg1="{ping.src.rc}" arg2="0"/>
    <equals arg1="{ping.dst.rc}" arg2="0"/>
  </and>
</condition>
<fail unless="run.ping" message="*** AGENTS NOT RUNNING ***"/>
</target>

<target name="fileTransfer" depends="ping" description="transfer file" if="run.ping">
  <fte:filecopy cmdqm="{cmd.qmgr}" src="{src}" dst="{dst}" jobname="{job.name}"
rcproperty="copy1.rc">
    <fte:filespec srcfilespec="{src.file1}" dstfile="{dst.file1}" overwrite="true"
checksum="MD5" conversion="binary"/>
  </fte:filecopy>
  <fte:filecopy cmdqm="{cmd.qmgr}" src="{src}" dst="{dst}" jobname="{job.name}"
rcproperty="copy2.rc">
    <fte:filespec srcfilespec="{src.file2}" dstfile="{dst.file2}" overwrite="true"
checksum="MD5" conversion="binary"/>
  </fte:filecopy>
  <condition property="file1.transfer.failed">
    <not><equals arg1="{copy1.rc}" arg2="0"/></not>
  </condition>
  <condition property="file2.transfer.failed">
    <not><equals arg1="{copy2.rc}" arg2="0"/></not>
  </condition>
</target>

<target name="email1" depends="fileTransfer" description="send email"
if="file1.transfer.failed">
  <mail mailhost="{mail.host}" mailport="20001" encoding="plain" subject="{job.name}
failed!">
    <from address="fteadmin@itso.com"/>
    <to address="{mail.to}"/>
    <message>Transfer of Inventory file FAILED: {job.name} from agent {src} to agent
{dst} has failed with return code: {copy1.rc}</message>
  </mail>
</target>

<target name="email2" depends="fileTransfer" description="send email"
if="file2.transfer.failed">
  <mail mailhost="{mail.host}" mailport="20001" encoding="plain" subject="{job.name}
failed!">
    <from address="fteadmin@itso.com"/>
    <to address="{mail.to}"/>
    <message>Transfer of trigger file FAILED: {job.name} from agent {src} to agent {dst}
has failed with return code: {copy2.rc}</message>
  </mail>
</target>

<target name="complete" depends="email1, email2"/>
</project>

```

Save this script at the following location, E:\INVENTORY\NEWYORK\transfer.xml, on the server store.newyork.itso.com.

concatenate.xml Ant script

ITSO Enterprise developed this Ant script to concatenate the inventory files that are transferred from the stores into a single inventory file. Example 6-54 illustrates the various Ant tasks that are put together in the script.

This Ant script consists of the following sections:

- ▶ Initialize the operational arguments
- ▶ Concatenate the inventory files
- ▶ Merge the files from the stores into one inventory file named `ITSO.INVENTORY.<yyyyMMdd>.csv`

Assess the transfer and, if it fails, send an email to the store admin.

Example 6-54 Ant script concatenate.xml for the warehouse

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns:fte="antlib:com.ibm.wmqfte.ant.taskdefs" default="complete">

<target name="init">
  <property name="in.file1"
value="E:\ITSO.INVENTORY.FILES\WAREHOUSE\ITSOCHI.INVENTORY.csv"/>
  <property name="in.file2"
value="E:\ITSO.INVENTORY.FILES\WAREHOUSE\ITSONEW.INVENTORY.csv"/>
  <property name="in.file3"
value="E:\ITSO.INVENTORY.FILES\WAREHOUSE\ITSOATL.INVENTORY.csv"/>
  <property name="in.file11"
value="E:\ITSO.INVENTORY.FILES\WAREHOUSE\ITSOCHI.INVENTORY.go"/>
  <property name="in.file22"
value="E:\ITSO.INVENTORY.FILES\WAREHOUSE\ITSONEW.INVENTORY.go"/>
  <property name="in.file33"
value="E:\ITSO.INVENTORY.FILES\WAREHOUSE\ITSOATL.INVENTORY.go"/>
  <tstamp/>
  <property name="out.file"
value="E:\ITSO.INVENTORY.FILES\WAREHOUSE\ITSO.INVENTORY.${DSTAMP}.csv"/>
</target>

<target name="concatenate" depends="init">
  <concat destfile="${out.file}" append="true">
    <fileset file="${in.file1}"/>
    <fileset file="${in.file2}"/>
    <fileset file="${in.file3}"/>
  </concat>
</target>

<target name="cleanup" depends="concatenate">
  <delete file="${in.file1}" failonerror="false"/>
  <delete file="${in.file2}" failonerror="false"/>
  <delete file="${in.file3}" failonerror="false"/>
  <delete file="${in.file11}" failonerror="false"/>
  <delete file="${in.file22}" failonerror="false"/>
  <delete file="${in.file33}" failonerror="false"/>
</target>

<target name="complete" depends="cleanup"/>
</project>
```

Save this script at the following location, E:\INVENTORY\WAREHOUSE\concatenate.xml, on the server warehouse.itso.com.

Configuring the monitoring tasks to start commands and scripts

You can configure resource monitors to call other commands from the monitoring agent, including executable programs, Ant scripts, or JCL jobs. First, we explain the resource monitor.

What is resource monitoring

By using the resource monitoring feature of WebSphere MQ File Transfer Edition, you can monitor a resource, for example, a queue or a directory. When a certain condition on this resource is satisfied, the resource monitor starts a task, such as a file transfer. You can create a resource monitor by using the **fteCreateMonitor** command.

A common scenario is to monitor a directory for the presence of a trigger file. An external application might process multiple files and place them in a known source directory. When the application has completed its processing, it indicates that the files are ready to be transferred, or otherwise acted upon, by placing a trigger file into a monitored location. The trigger file can be detected by a WebSphere MQ File Transfer Edition monitor, and the transfer of those files from the source directory to another WebSphere MQ File Transfer Edition agent is initiated. In the case of the ITSO Enterprise stores, the stores place a *.go file, which helps the resource monitor understand that the inventory file is now ready to be transferred to the warehouse. Example 6-55 illustrates the syntax for the **fteCreateMonitor** command.

Example 6-55 The syntax for fteCreateMonitor command

```
fteCreateMonitor [-p ConfigurationOptions]
                 -ma agent name [-mm agent queue manager]
                 (-md directory | -mq queue)
                 -mn monitor name
                 -mt task filename
                 [-pi poll interval]
                 [-pt pattern type]
                 [-pu poll units]
                 [-rl recursion level]
                 [-bs maximum batch size]
                 [-tr trigger condition]
                 [-x exclude pattern]
                 [-jn job name]
                 [-dvs substitution variable default values]
```

To get more information about all of the parameters for the **fteCreateMonitor** command, refer to this website:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.doc/create_monitor_cmd.htm

Creating monitor tasks

We can create the task definition XML document in one of the following ways:

- ▶ Create the task definition XML document manually according to the FileTransfer.xsd schema.
- ▶ Edit the XML document that was generated by the **fteCreateTransfer** command using the **-gt** parameter as the basis for your task definition.

To read more about creating monitoring tasks, refer to this website:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.doc/configuring_monitor_tasks.htm

We create the tasks manually following the schema of `FileTransfer.xsd`. We create a file named `inventoryTransferTask.xml` for the Atlanta, Chicago, and NEWYORK stores. These task files make a call to the Ant script `transfer.xml` that was created previously for the ITSO Enterprise stores.

Example 6-56 illustrates the monitor task file for the Atlanta store. Save this file in `E:\INVENTORY\ATLANTA\inventoryTransferTask.xml` on the server `store.atlanta.itso.com`.

Example 6-56 Creating inventoryTransferTask for the ITSO Enterprise Atlanta store

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
<managedCall>
  <originator>
    <hostName>store.atlanta.itso.com</hostName>
    <userID>wmquser</userID>
  </originator>
  <agent QMgr="STORE.ATLANTA.QM" agent="STORE.ATLANTA.QM"/>
  <reply QMGR="STORE.ATLANTA.QM">reply</reply>
  <transferSet priority="1">
    <call>
      <command name="transfer.xml" type="antscript" retryCount="0"
retryWait="0" successRC="0"/>
    </call>
  </transferSet>
</managedCall>
</request>
```

Example 6-57 illustrates the monitor task file for Chicago store. Save this file in `E:\INVENTORY\CHICAGO\inventoryTransferTask.xml` on the server `store.chicago.itso.com`.

Example 6-57 Creating inventoryTransferTask for the ITSO Enterprise Chicago store

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
<managedCall>
  <originator>
    <hostName>store.chicago.itso.com</hostName>
    <userID>wmquser</userID>
  </originator>
  <agent QMgr="STORE.CHICAGO.QM" agent="STORE.CHICAGO.QM"/>
  <reply QMGR="STORE.CHICAGO.QM">reply</reply>
  <transferSet priority="1">
    <call>
      <command name="transfer.xml" type="antscript" retryCount="0"
retryWait="0" successRC="0"/>
    </call>
  </transferSet>
</managedCall>
</request>
```

Example 6-58 illustrates the monitor task file for the New York store. Save this file in E:\INVENTORY\NEWYORK\inventoryTransferTask.xml on the server store.newyork.itso.com.

Example 6-58 Creating inventoryTransferTask for the ITSO Enterprise Newyork store

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.00" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
<managedCall>
  <originator>
    <hostName>store.newyork.itso.com</hostName>
    <userID>wmquser</userID>
  </originator>
  <agent QMgr="STORE.NEWYORK.QM" agent="STORE.NEWYORK.QM"/>
  <reply QMGR="STORE.NEWYORK.QM">reply</reply>
  <transferSet priority="1">
    <call>
      <command name="transfer.xml" type="antscript" retryCount="0"
retryWait="0" successRC="0"/>
    </call>
  </transferSet>
</managedCall>
</request>
```

For the warehouse monitor, create a task file called inventoryConcatenateTask.xml. This task file makes a call to the Ant script concatenate.xml, which was created previously for the ITSO Enterprise warehouse.

Example 6-59 illustrates the monitor task file for the ITSO Enterprise warehouse. Save this file in the E:\INVENTORY\WAREHOUSE\inventoryConcatenateTask.xml on the server warehouse.itso.com.

Example 6-59 Creating inventoryConcatenateTask.xml for the ITSO Enterprise warehouse

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.00" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
<managedCall>
  <originator>
    <hostName>warehouse.itso.com</hostName>
    <userID>wmquser</userID>
  </originator>
  <agent QMgr="WAREHOUSE.AGENT.QM" agent="WAREHOUSE.AGENT.QM"/>
  <reply QMGR="WAREHOUSE.AGENT.QM">reply</reply>
  <transferSet priority="1">
    <call>
      <command name="concatenate.xml" type="antscript" retryCount="0"
retryWait="0" successRC="0"/>
    </call>
  </transferSet>
</managedCall>
</request>
```

In order for the File Transfer Edition agents to recognize the location of the Ant scripts, add commandPath in the agent.properties file for all the store and warehouse agents. To read more about commandPath properties for the File Transfer Edition agents, refer to this website:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.doc/comm and_path.htm

“Updated agent.properties for File Transfer Edition agent STORE.ATLANTA.QM” on page 279 illustrates the updated agent .properties file for the File Transfer Edition agent STORE.ATLANTA.QM.

```
1 agent.properties
#
#Mon Jun 27 07:36:14 EDT 2011
agentQMgr=STORE.ATLANTA.QM
agentQMgrPort=1417
agentDesc=Agent for ITSO Atlanta Store
agentQMgrHost=store.atlanta.itso.com
agentQMgrChannel=ITSO.SSL.SVRCONN
agentName=STORE.ATLANTA.QM
commandPath=E:\\Scripts\\ATLANTA
```

Figure 6-26 Updated agent.properties for File Transfer Edition agent STORE.ATLANTA.QM

Similarly, update the agent .properties files for the other agents that are listed in Table 6-3.

Table 6-3 Update agent.properties files with commandPath

Agent name	commandPath value in agent.properties file
STORE.ATLANTA.QM	E:\\Scripts\\ATLANTA
STORE.CHICAGO.QM	E:\\Scripts\\CHICAGO
STORE.NEWYORK.QM	E:\\Scripts\\NEWYORK
WAREHOUSE.AGENT.QM	E:\\Scripts\\WAREHOUSE

After you have changed the agent .properties files for the agents, restart the agents using the **fteStopAgent** and **fteStartAgent** commands.

Creating the monitor

Until now, we have created Ant scripts and monitor tasks that execute these Ant scripts. We now create resource monitors, which, based on an event, will execute a monitor task. We use the **fteCreateMonitor** command to create the resource monitor to monitor the daily inventory files that are created on the stores and the warehouse.

Example 6-60 illustrates the monitor for the ITSO Enterprise Atlanta store. This monitor polls for the existence of and the changes on the trigger file `ITS0ATL.INVENTORY.go` in the directory `E:\\ITSO.INVENTORY.FILES\\ATLANTA` of the server `store.atlanta.itso.com`.

Example 6-60 Monitor for ITSO Enterprise Atlanta store

```
fteCreateMonitor
-ma STORE.ATLANTA.QM
-mm STORE.ATLANTA.QM
-md E:\\ITSO.INVENTORY.FILES\\ATLANTA
-mn ATLANTA.DAILY.INVENTORY.MONITOR
-mt E:\\Scripts\\ATLANTA\\inventoryTransferTask.xml
-pi 1 -pu minutes
```

```
-tr match,ITSOATL.INVENTORY.go
```

Example 6-61 illustrates the monitor for the ITSO Enterprise Chicago store. This monitor polls for the existence of and the changes on the trigger file `ITSOCHI.INVENTORY.go` in the directory `E:\ITSO.INVENTORY.FILES\CHICAGO` of the server `store.chicago.itso.com`.

Example 6-61 Monitor for ITSO Enterprise Chicago store

```
fteCreateMonitor
-ma STORE.CHICAGO.QM
-mm STORE.CHICAGO.QM
-md E:\ITSO.INVENTORY.FILES\CHICAGO
-mn CHICAGO.DAILY.INVENTORY.MONITOR
-mt E:\Scripts\CHICAGO\inventoryTransferTask.xml
-pi 1 -pu minutes
-tr match,ITSOCHI.INVENTORY.go
```

Example 6-62 illustrates the monitor for the ITSO Enterprise New York store. This monitor polls for the existence of and the changes on the trigger file `ITSONEW.INVENTORY.go` in the directory `E:\ITSO.INVENTORY.FILES\NEWYORK` of the server `store.newyork.itso.com`.

Example 6-62 Monitor for ITSO Enterprise NEWYORK store

```
fteCreateMonitor
-ma STORE.NEWYORK.QM
-mm STORE.NEWYORK.QM
-md E:\ITSO.INVENTORY.FILES\NEWYORK
-mn NEWYORK.DAILY.INVENTORY.MONITOR
-mt E:\Scripts\NEWYORK\inventoryTransferTask.xml
-pi 1 -pu minutes
-tr match,ITSONEW.INVENTORY.go
```

Example 6-63 illustrates the monitor for the ITSO Enterprise warehouse. This monitor polls for the existence of and changes on the trigger file `*.go` in the directory `E:\ITSO.INVENTORY.FILES\WAREHOUSE` of the server `warehouse.itso.com`.

Example 6-63 Monitor for the ITSO Enterprise warehouse

```
fteCreateMonitor
-ma WAREHOUSE.AGENT.QM
-mm WAREHOUSE.AGENT.QM
-md E:\ITSO.INVENTORY.FILES\WAREHOUSE
-mn WAREHOUSE.DAILY.INVENTORY.MONITOR
-mt E:\Scripts\WAREHOUSE\inventoryConcatenateTask.xml
-pi 1 -pu minutes
-tr match,*.go
```

Polling interval: The polling Interval on the monitors is set to 1 minute for testing purposes, which might be an extremely low value for production systems.

Use the `fteListMonitors` command to list all of the existing resource monitors in a WebSphere MQ File Transfer Edition network by using the command line. Example 6-64 on page 281 illustrates the syntax for the command `fteListMonitors`.

Example 6-64 The syntax for the command fteListMonitors

```
fteListMonitors [-p ConfigurationOptions]
                [-ma AgentName] [-mn MonitorName] [-v]
```

After the monitors are created, execute the **fteListMonitors** command on any store or warehouse server. Example 6-65 illustrates the list of the resource monitors that have been created for the ITSO Enterprise stores and the warehouse.

Example 6-65 List of resource monitors that have been created on the ITSO stores and the warehouse

```
E:\>fteListMonitors -v
5655-U80, 5724-R10 Copyright IBM Corp. 2008, 2011. ALL RIGHTS RESERVED
Monitor Information:
  Name:          ATLANTA.DAILY.INVENTORY.MONITOR
  Agent:         STORE.ATLANTA.QM
  Status:        Started
  Resource Type: Directory
  Resource:      E:\ITSO.INVENTORY.FILES\ATLANTA
  Poll interval: 1 minutes
  Batch size:    1
  Condition:     Match
  Pattern:       ITSOATL.INVENTORY.go (wildcard)
Monitor Information:
  Name:          CHICAGO.DAILY.INVENTORY.MONITOR
  Agent:         STORE.CHICAGO.QM
  Status:        Started
  Resource Type: Directory
  Resource:      E:\ITSO.INVENTORY.FILES\CHICAGO
  Poll interval: 1 minutes
  Batch size:    1
  Condition:     Match
  Pattern:       ITSOCHI.INVENTORY.go (wildcard)
Monitor Information:
  Name:          NEWYORK.DAILY.INVENTORY.MONITOR
  Agent:         STORE.NEWYORK.QM
  Status:        Started
  Resource Type: Directory
  Resource:      E:\ITSO.INVENTORY.FILES\NEWYORK
  Poll interval: 1 minutes
  Batch size:    1
  Condition:     Match
  Pattern:       ITSONEW.INVENTORY.go (wildcard)
Monitor Information:
  Name:          WAREHOUSE.DAILY.INVENTORY.MONITOR
  Agent:         WAREHOUSE.AGENT.QM
  Status:        Started
  Resource Type: Directory
  Resource:      E:\ITSO.INVENTORY.FILES\WAREHOUSE
  Poll interval: 1 minutes
  Batch size:    1
  Condition:     Match
  Pattern:       *.go (wildcard)
```

You can also view details about and administer the resource monitor from WebSphere MQ Explorer.

Archiving and transferring the file for inventory processing

Using the resource monitor that you created in this section, you can transfer the daily inventory file from the stores to the warehouse. Also, the warehouse monitor merges the separate store inventory files into one large enterprise inventory file.

However, the concatenated file must be moved to the Inventory processing system and a copy must be stored in the archive. Based on the business requirements, the concatenated file must be transferred to the Inventory processing system at 2:00 a.m. and a copy must be stored in the archive repository at 2:30 a.m. everyday.

Example 6-66 illustrates the command to create a scheduled transfer of ITSO Enterprise inventory file to Inventory processing system at 2:00 AM everyday.

Example 6-66 Scheduled inventory file transfer to the Inventory processing system

```
fteCreateTransfer
-sa WAREHOUSE.AGENT.QM
-sm WAREHOUSE.AGENT.QM
-da WAREHOUSE.AGENT.QM
-dm WAREHOUSE.AGENT.QM
-dd E:\ITSO.INVENTORY.FILES\WAREHOUSE\INVIT
-t binary -de error
-ss 02:00 -oi days
E:\ITSO.INVENTORY.FILES\WAREHOUSE\ITSO.INVENTORY.*.csv
```

Example 6-67 illustrates the command to create a scheduled transfer of the ITSO Enterprise inventory file to the archive repository at 2:30 a.m. everyday.

Example 6-67 Scheduled inventory file transfer to the archive repository

```
fteCreateTransfer
-sa WAREHOUSE.AGENT.QM
-sm WAREHOUSE.AGENT.QM
-da WAREHOUSE.AGENT.QM
-dm WAREHOUSE.AGENT.QM
-dd E:\ITSO.INVENTORY.FILES\WAREHOUSE\ARCHIVE
-t binary -de error
-sd delete
-ss 02:30 -oi days
E:\ITSO.INVENTORY.FILES\WAREHOUSE\ITSO.INVENTORY.*.csv
```

You can use the **ftelistScheduledTransfers** command to list the scheduled transfers. Example 6-68 illustrates the scheduled transfers for the ITSO Enterprise inventory file.

Example 6-68 Scheduled transfer of the ITSO Enterprise inventory file

```
C:> ftelistScheduledTransfers
5655-U80, 5724-R10 Copyright IBM Corp. 2008, 2011. ALL RIGHTS RESERVED
Schedule Identifier: 4
Source Agent Name: WAREHOUSE.AGENT.QM
Source File Name: E:\ITSO.INVENTORY.FILES\WAREHOUSE\ITSO.INVENTORY.*.csv
Conversion Type: binary
Destination File Name: E:\ITSO.INVENTORY.FILES\WAREHOUSE\INVIT
Destination Agent Name: WAREHOUSE.AGENT.QM
Schedule Start Time: 2011-06-28T02:00-0400
Next Transfer: 2011-06-29T02:00-0400
Schedule Time Base: admin
```


Repeat Interval: days
Repeat Frequency: 1

Schedule Identifier: 5
Source Agent Name: WAREHOUSE.AGENT.QM
Source File Name: E:\ITSO.INVENTORY.FILES\WAREHOUSE\ITSO.INVENTORY.*.csv
Conversion Type: binary
Destination File Name: E:\ITSO.INVENTORY.FILES\WAREHOUSE\ARCHIVE
Destination Agent Name: WAREHOUSE.AGENT.QM
Schedule Start Time: 2011-06-28T02:30-0400
Next Transfer: 2011-06-29T02:30-0400
Schedule Time Base: admin
Repeat Interval: days
Repeat Frequency: 1

6.6.3 Testing task automation

To test this business scenario, we perform a success and a failure use case. The success use case demonstrates that after the inventory files are created in the stores, they are automatically moved to the warehouse. Later, at the warehouse, the inventories are merged, archived, and sent for processing to the Inventory processing system.

The failure use case demonstrates that if there is a failure while transferring a file from a store to the warehouse, the problem is immediately reported to the store admin via email from the File Transfer Edition admin email ID, for example, fteadmin@itso.com.

Success use case

Create the inventory file and the trigger file on the Atlanta store. Figure 6-27 illustrates the files that were created for transfer at the ITSO Enterprise Atlanta store. The resource monitors track the modification time stamp of the trigger file and, based on that time stamp, trigger the monitoring tasks. After the trigger file is created, the resource monitor ATLANTA.DAILY.INVENTORY.MONITOR identifies that within a minute and starts the monitoring task.



Name ^	Date modified	Type
 ITSOATL.INVENTORY	6/29/2011 11:24 PM	CSV File
 ITSOATL.INVENTORY	6/29/2011 11:24 PM	GO File

Figure 6-27 Inventory and trigger files for the ITSO Enterprise Atlanta store

Figure 6-28 illustrates the inventory and trigger files that were created for the ITSO Enterprise Chicago store.



Name ^	Date modified	Type
 ITSOCHI.INVENTORY	6/29/2011 11:25 PM	CSV File
 ITSOCHI.INVENTORY	6/29/2011 11:25 PM	GO File

Figure 6-28 Inventory and trigger files for the ITSO Enterprise Chicago store

Figure 6-29 on page 284 illustrates the inventory and trigger files that were created for the ITSO Enterprise NEWYORK store.

Name ^	Date modified	Type
ITSONEW.INVENTORY	6/29/2011 11:25 PM	CSV File
ITSONEW.INVENTORY	6/29/2011 11:25 PM	GO File

Figure 6-29 Inventory and trigger files for the NEWYORK store

The resource monitors, CHICAGO.DAILY.INVENTORY.MONITOR and NEWYORK.DAILY.INVENTORY.MONITOR, identify the trigger file changes within 1 minute and create the file transfers to the warehouse agent.

Figure 6-30 illustrates the status of the completed transfers. For each store, there are three entries in the logs: one entry for starting the monitor task and the second and third entries for transferring the inventory and trigger files.

Based on the presence of the trigger files that are transferred to the warehouse, the warehouse resource monitor invokes the concatenation task and merges the files into a single inventory file.

	Source	Destination	Completion State	Started (America/New_York)
+	STORE.ATLANTA.QM		Successful	6/29/11 11:25:20 PM EDT
+	STORE.ATLANTA.QM	WAREHOUSE.AGENT.QM	Successful	6/29/11 11:25:21 PM EDT
+	STORE.ATLANTA.QM	WAREHOUSE.AGENT.QM	Successful	6/29/11 11:25:22 PM EDT
+	STORE.CHICAGO.QM		Successful	6/29/11 11:25:25 PM EDT
+	STORE.CHICAGO.QM	WAREHOUSE.AGENT.QM	Successful	6/29/11 11:25:27 PM EDT
+	STORE.CHICAGO.QM	WAREHOUSE.AGENT.QM	Successful	6/29/11 11:25:27 PM EDT
-	WAREHOUSE.AGENT.QM		Successful	6/29/11 11:25:50 PM EDT
	[call] E:\Scripts\WAREHOUSE\concatenate.xml		Successful	
-	WAREHOUSE.AGENT.QM		Successful	6/29/11 11:25:50 PM EDT
	[call] E:\Scripts\WAREHOUSE\concatenate.xml		Successful	
+	STORE.NEWYORK.QM		Successful	6/29/11 11:26:55 PM EDT
+	STORE.NEWYORK.QM	WAREHOUSE.AGENT.QM	Successful	6/29/11 11:26:57 PM EDT
+	STORE.NEWYORK.QM	WAREHOUSE.AGENT.QM	Successful	6/29/11 11:26:57 PM EDT
-	WAREHOUSE.AGENT.QM		Successful	6/29/11 11:27:50 PM EDT
	[call] E:\Scripts\WAREHOUSE\concatenate.xml		Successful	

Figure 6-30 Transfer log for the files that are transferred from the stores to the warehouse

After the completion of the concatenation task, the warehouse has a concatenated inventory file with the name ITS0.INVENTORY.<yyyyMMdd>.csv. Figure 6-31 illustrates the daily ITS0 Enterprise inventory file that was created for 29 June 2011.

Name ^	Date modified	Type
ARCHIVE	6/28/2011 5:11 PM	File folder
INVIT	6/28/2011 5:10 PM	File folder
ITS0.INVENTORY.20110629	6/29/2011 11:27 PM	CSV File

Figure 6-31 Daily inventory file that was created in the warehouse

At 2:00 a.m., the inventory file must be copied into the INVIT folder to be processed later by the Inventory processing system. This copy is done through the scheduled transfer that is illustrated in Example 6-66 on page 282. Figure 6-32 shows the inventory file that was copied to the INVIT folder at 2:00 a.m.

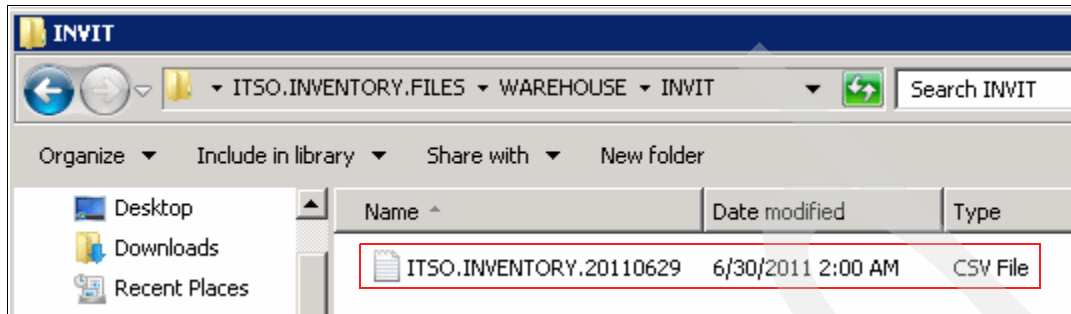


Figure 6-32 Daily inventory file that was copied for the Inventory processing system

At 2:30 a.m., the inventory file must be moved into the ARCHIVE folder and is stored for future reference. This move is done through the scheduled transfer that is illustrated in Example 6-67 on page 282. Figure 6-33 shows the inventory file that is moved to the ARCHIVE folder at 2:30 a.m.

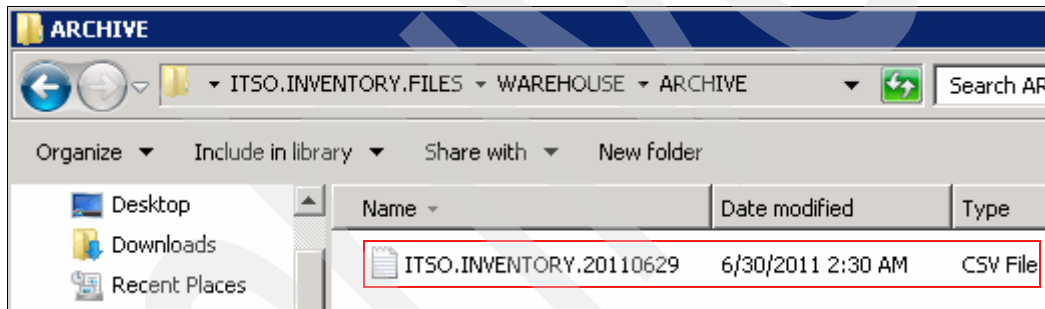


Figure 6-33 Daily inventory file moved into ARCHIVE repository

We have shown you how the end-to-end inventory file transfer tasks are automated.

Failure use case

Back up the Ant script `transfer.xml` that was created for the Atlanta store as `transfer.xml.bak`. Create a new `transfer.xml` file, as illustrated in Example 6-69. The only difference in this script is that it transfers only the inventory file and does not overwrite the file at destination.

Therefore, if the file exists at the destination, the transfer fails and causes the store admin to be alerted by email.

Example 6-69 New `transfer.xml` created for Atlanta store to create a failure

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns:fte="antlib:com.ibm.wmqfte.ant.taskdefs" default="complete">

  <target name="init" description="initialise task properties">
    <property name="cmd.qmgr"
value="ITSO.FILE.COMMAND.QM@fileadmin.itso.com@1415@ITSO.SSL.SVRCONN"/>
    <property name="src" value="STORE.ATLANTA.QM@STORE.ATLANTA.QM"/>
  </target>
</project>
```

```

        <property name="dst"
value="WAREHOUSE.AGENT.QM@WAREHOUSE.AGENT.QM"/>
        <property name="mail.host" value="smtp.itso.com"/>
        <property name="mail.to" value="atlantaadmin@itso.com"/>
        <property name="src.file1"
value="E:\ITSO.INVENTORY.FILES\ATLANTA\ITSOATL.INVENTORY.csv"/>
        <property name="dst.file1"
value="E:\ITSO.INVENTORY.FILES\WAREHOUSE\ITSOATL.INVENTORY.csv"/>
        <fte:uuid property="job.name" length="8" prefix="Transfer Daily
Inventory "/>
    </target>

    <target name="ping" depends="init" description="check agents">
        <fte:ping cmdqm="{cmd.qmgr}" agent="{src}"
rcproperty="ping.src.rc" timeout="10"/>
        <fte:ping cmdqm="{cmd.qmgr}" agent="{dst}"
rcproperty="ping.dst.rc" timeout="10"/>
        <condition property="run.ping">
            <and>
                <equals arg1="{ping.src.rc}" arg2="0"/>
                <equals arg1="{ping.dst.rc}" arg2="0"/>
            </and>
        </condition>
        <fail unless="run.ping" message="*** AGENTS NOT RUNNING ***"/>
    </target>

    <target name="fileTransfer" depends="ping" description="transfer file"
if="run.ping">
        <fte:filecopy cmdqm="{cmd.qmgr}" src="{src}" dst="{dst}"
jobname="{job.name}" rcproperty="copy1.rc">
            <fte:filespec srcfilespec="{src.file1}"
dstfile="{dst.file1}" checksum="MD5" conversion="binary"/>
        </fte:filecopy>
        <condition property="file1.transfer.failed">
            <not><equals arg1="{copy1.rc}" arg2="0"/></not>
        </condition>
    </target>

    <target name="email1" depends="fileTransfer" description="send email"
if="file1.transfer.failed">
        <mail mailhost="{mail.host}" mailport="20001" encoding="plain"
subject="{job.name} failed! ">
            <from address="fteadmin@itso.com"/>
            <to address="{mail.to}"/>
            <message>Transfer of Inventory file FAILED: {job.name}
from agent {src} to agent {dst} has failed with return code:
{copy1.rc}</message>
        </mail>
    </target>

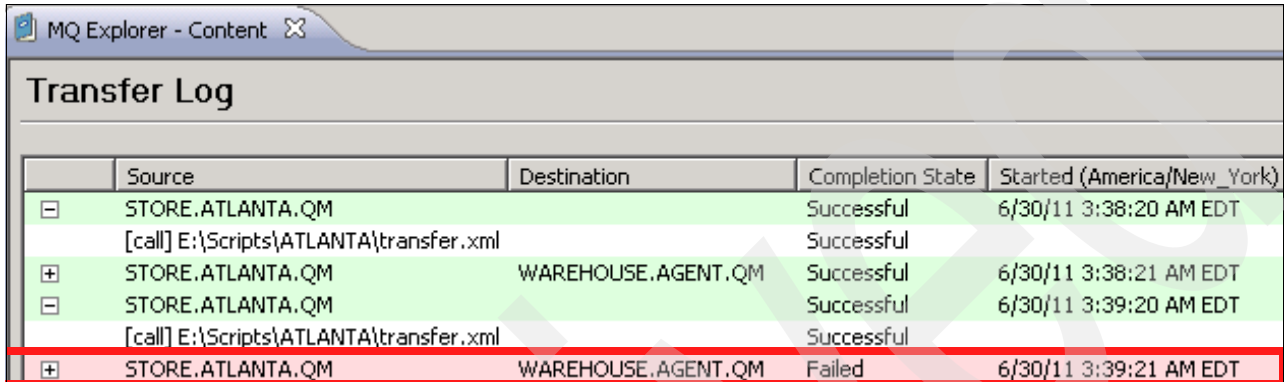
    <target name="complete" depends="email1"/>
</project>

```

Steps to create a failure

Use these steps to create a failure:

1. Modify and save the trigger file `ITSOATL.INVENTORY.go`, which will create a new transfer request. The inventory file `ITSOATL.INVENTORY.csv` will be moved to the warehouse.
2. Modify the trigger file again; however, this time, the file transfer will fail, because it cannot overwrite the file at the destination.
3. Verify the transfer log in WebSphere MQ Explorer. Verify that the first transfer was successful and that the second transfer failed. See Figure 6-34.



	Source	Destination	Completion State	Started (America/New_York)
[-]	STORE.ATLANTA.QM		Successful	6/30/11 3:38:20 AM EDT
	[call] E:\Scripts\ATLANTA\transfer.xml		Successful	
[+]	STORE.ATLANTA.QM	WAREHOUSE.AGENT.QM	Successful	6/30/11 3:38:21 AM EDT
[-]	STORE.ATLANTA.QM		Successful	6/30/11 3:39:20 AM EDT
	[call] E:\Scripts\ATLANTA\transfer.xml		Successful	
[+]	STORE.ATLANTA.QM	WAREHOUSE.AGENT.QM	Failed	6/30/11 3:39:21 AM EDT

Figure 6-34 Transfer log for the task automation failure use case

Email to the store admin

Because the inventory file transfer has failed, the Atlanta store admin will immediately receive an email from `ftheadmin@itso.com`, stating the details about the failure.

Figure 6-35 illustrates the notification email that was sent to the Atlanta store admin, stating the failure and the reason code.

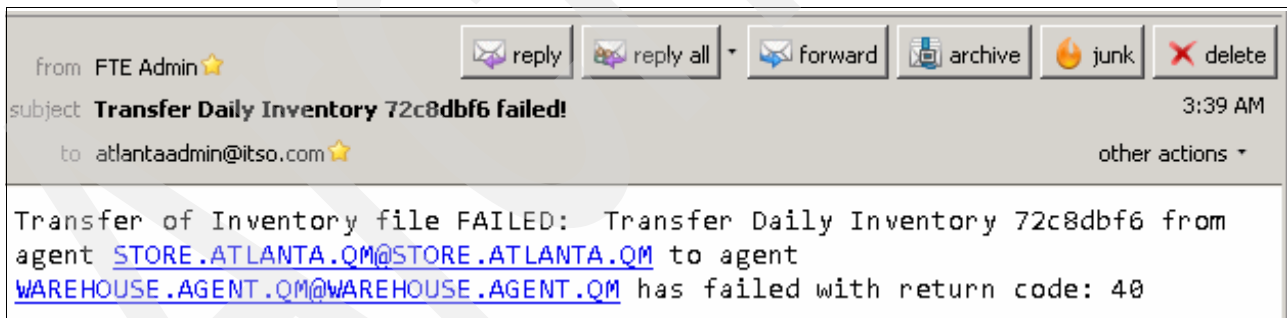


Figure 6-35 Email sent to Atlanta store admin stating file transfer failures

With the completion of the success and failure tests on the task automation, all the task automation requirements for ITSO Enterprise were implemented using WebSphere MQ File Transfer Edition.

6.7 Business scenario 3

In this business scenario, we demonstrate the capabilities of the WebSphere MQ File Transfer Edition events auditing and archiving feature.

6.7.1 Requirement

ITSO Enterprise conducts a weekly audit on the inventory files that are transferred from the ITSO Enterprise stores to the ITSO Enterprise warehouse. Later, ITSO Enterprise must generate a report on the inventory files that are transferred from the ITSO Enterprise stores to the ITSO Enterprise warehouse. Currently, ITSO Enterprise does not have any procedure to archive the file transfer events and later prepare an audit report on a timely basis. The company immediately needs a file transfer event auditing and archiving functionality.

6.7.2 Solution

WebSphere MQ File Transfer Edition provides an additional software package called the *database logger* to store audit information in an SQL database. When the File Transfer Edition transfers files, it publishes information about its actions to a topic on the coordination queue manager. You can use the database logger to copy this information into an SQL table for analysis and auditing purposes.

There are two versions of the database logger: a stand-alone Java Platform, Standard Edition (JSE) application and a Java Platform, Enterprise Edition (JEE) application. The stand-alone database logger is a Java application that you install on a system that hosts the coordination queue manager and the database. The stand-alone database logger connects to the coordination queue manager using WebSphere MQ bindings, and to a DB2 or Oracle database using the type 2 Java Database Connectivity (JDBC) driver. You require this type of connection, because the database logger uses the queue manager's extended architecture (XA) support to coordinate a global transaction over both the queue manager and database, protecting the data.

Installing the database logger

The database logger is an optional component of File Transfer Edition, and if you want to use it, you can get it from the Remote Tools CD.

the stand-alone database logger requires the following products:

- ▶ The WebSphere MQ server installation must be WebSphere MQ 7.0 Fix Pack 7.0.1.0 or higher
- ▶ The operating system must be supported for the Server version of WebSphere MQ File Transfer Edition
- ▶ The operating system and the database must be supported for WebSphere MQ to act as a Transaction Coordinator for WebSphere MQ classes for Java applications (via JDBC XA).
- ▶ You must have one of the following supported databases:
 - ▶ DB2 9.1
 - ▶ DB2 9.5
 - ▶ DB2 9.7 with Fix Pack 1 or later
 - ▶ Oracle 10g or Oracle 10.2
 - ▶ Oracle 11g

During the installation, choose this component in the “Select Set of Features to install” window or click **Complete installation**. Figure 6-36 on page 289 illustrates the database logger installation options.

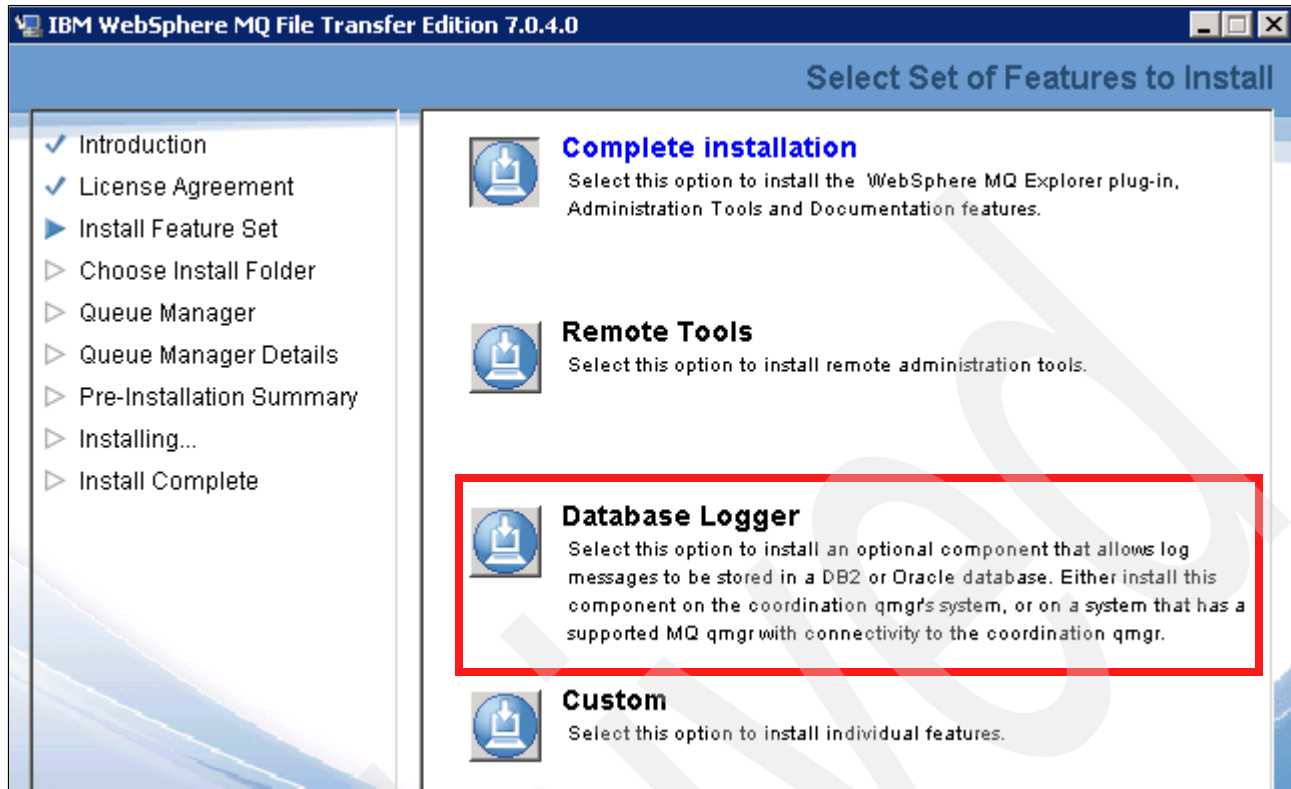


Figure 6-36 Database Logger installation window

Configuring the database logger

Before you can start the database logger command, you must complete these tasks:

1. Create the database and tables with SQL scripts.
2. Set the user permissions.
3. Configure the transaction support in the queue manager.
4. Edit the `databaselogger.properties` file.

Figure 6-37 illustrates the XA configuration that is performed on the coordination queue manager `ITSO.FILE.COORDINATION.QM`.

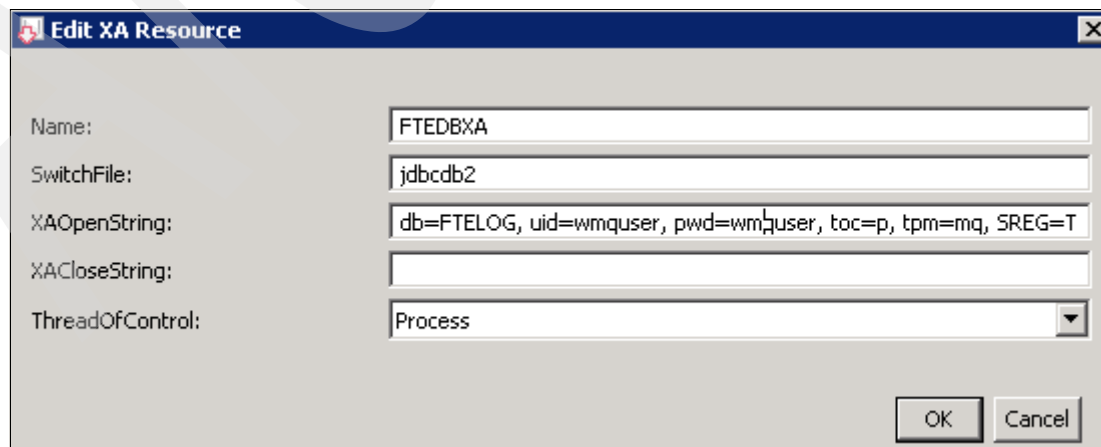


Figure 6-37 XA configuration on `ITSO.FILE.COORDINATION.QM`

Possible error message: You might receive the following error while starting the queue manager on a Microsoft Windows server, after the XA configurations have been saved.

```
WebSphere MQ could not load the XA switch load file for resource manager
'FTEDBXA'.
```

An error has occurred loading the XA switch file jdbcd2. If the error occurred during start-up, the queue manager will terminate. At all other times, the queue manager will continue without this resource manager, which means that it will no longer be able to participate in global transactions. The queue manager will retry the load of the switch file at regular intervals so that the resource manager will be able to participate again if the load problem is resolved.

Look for a previous message outlining the reason for the load failure. Message AMQ6175 is issued if the load failed because of a system error. If you see this message, follow the guidance that is given in message AMQ6175 to resolve the problem. In the absence of prior messages or IBM First Failure Support Technology (FFST™) information that relates to this problem, check that the name of the switch load file is correct and that it is present in a directory from which it can be dynamically loaded by the queue manager. The easiest method of checking is to define the switch load file as a fully qualified name. Note that if the queue manager is still running, it will need to be restarted so that any changes made to its configuration data can be picked up.

To start the queue manager, use the `strmqm -si <Queue Manager>` command.

Figure 6-38 illustrates the `databaselogger.properties` configuration in the ITSO Enterprise environment.

```
1 databaselogger.properties
wmqfte.queue.manager=ITSO.FILE.COORDINATION.QM
wmqfte.database.name=FTELOG
wmqfte.database.driver=C:\\db2\\java\\db2jcc.jar
```

Figure 6-38 The `databaselogger.properties` file for the File Transfer Edition database logger

For more configuration information, see “Using the database logger” in the WebSphere MQ File Transfer Edition Information Center, which is located at this website:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.doc/dl_install_standalone.htm

FTELOG: The name of the database that is created to use for ITSO Enterprise File Transfer Edition database logger is **FTELOG**, on DB2 V9.7. This database exists on the server `fileadmin.itso.com`.

Starting the database logger

After all of the configurations are ready, you can start the database logger using the `fteStartDatabaseLogger` command. By default, the database logger runs in the background, and it places its output in a file in the logs directory. If you want to run the database logger in the foreground, and have its output sent to the console and to the log file, add the `-F` switch to the `fteStartDatabaseLogger` command. Example 6-70 on page 291 illustrates the syntax for this command.

Example 6-70 The syntax for the fteStartDatabaseLogger command

```
fteStartDatabaseLogger [-p ConfigurationOptions] [-F  
                        [-b bits] [PropertiesFile]
```

To get more information about all the parameters of the **fteStartDatabaseLogger** command, see this website:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.doc/start_db_logger_cmd.htm

Because we run this setup on a Microsoft Windows Server 2008 R2 64-bit server, we use the command that is shown in Example 6-71 to start the database logger. After the database logger is started, it runs as a 64-bit Java process using 64-bit DB2 client libraries to connect the DB2 database FTELOG.

Example 6-71 The fteStartDatabaseLogger command

```
fteStartDatabaseLogger -b 64
```

The database logger is now installed, configured, and started. In 6.7.3, “Testing the file transfer audit feature” on page 291, we will test the file transfer auditing feature of WebSphere MQ File Transfer Edition.

6.7.3 Testing the file transfer audit feature

When the file transfers are running, the auditing and logging information is received and stored in a database by the database logger. In the File Transfer Edition log database, there are tables for basic transfers, scheduled transfers, file trigger conditions, metadata, and transfer calling actions:

- ▶ AUTH_EVENT
- ▶ CALL
- ▶ CALL_ARGUMENT
- ▶ CALL_REQUEST
- ▶ CALL_RESULT
- ▶ FILE_SPACE_ENTRY
- ▶ METADATA
- ▶ MONITOR
- ▶ MONITOR_ACTION
- ▶ MONITOR_EXIT_RESULT
- ▶ MONITOR_METADATA
- ▶ SCHEDULE
- ▶ SCHEDULE_ACTION
- ▶ SCHEDULE_SPEC
- ▶ SCHEDULE_ITEM
- ▶ TRANSFER
- ▶ TRANSFER_CALLS
- ▶ TRANSFER_CD_NODE
- ▶ TRANSFER_CORRELATOR
- ▶ TRANSFER_EVENT
- ▶ TRANSFER_EXIT
- ▶ TRANSFER_ITEM
- ▶ TRANSFER_STATS
- ▶ TRIGGER_CONDITION

To get more details about the tables of the File Transfer Edition database logger, see this website:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.doc/dl_tables.htm

The database logger helps to harden the transfer history information in the DB2 or Oracle database. In this scenario, we used DB2 V9.7 as the logger database. However, it does not provide a web or any other interface to display these records when users need to search the history transfer information.

Creating a file transfer request

Create a file transfer request by placing the inventory and trigger files in the File Transfer Edition agent monitoring location:

1. Place the `ITSOATL.INVENTORY.csv` and `ITSOATL.INVENTORY.go` files in the directory `E:\ITSO.INVENTORY.FILES\ATLANTA` of the server `store.atlanta.itso.com`.
2. Place the `ITSOCHI.INVENTORY.csv` and `ITSOCHI.INVENTORY.go` files in the directory `E:\ITSO.INVENTORY.FILES\CHICAGO` of the server `store.chicago.itso.com`.
3. Place the `ITSONEW.INVENTORY.csv` and `ITSONEW.INVENTORY.go` files in the directory `E:\ITSO.INVENTORY.FILES\NEWYORK` of the server `store.newyork.itso.com`.

After the presence of the trigger file is determined by the File Transfer Edition monitors, the monitoring tasks schedule an inventory file transfer and trigger file transfer from each store to the warehouse. Each file transfer generates events. Through the coordination queue manager, the File Transfer Edition database logger updates the various database tables with relevant event information.

Figure 6-39 illustrates the file transfer events that are archived in the database `FTELOG` that is used by the File Transfer Edition database logger. It also shows events for the two files that were transferred from three stores, which were successfully sent to the warehouse.

TRANSFER_ID	JOB_NAME	STA...	COMPLETE_ID	RESULTCODE	R
414d51204954...	Transfer Daily ...	540	541	0 BF	
414d51204954...	Transfer Daily ...	542	543	0 BF	
414d51204954...	Transfer Daily ...	544	545	0 BF	
414d51204954...	Transfer Daily ...	546	547	0 BF	
414d51204954...	Transfer Daily ...	548	549	0 BF	
414d51204954...	Transfer Daily ...	550	551	0 BF	

Figure 6-39 File transfer events archived in database `FTELOG`

This procedure to gather information to prepare a report can be tedious for users. Hence, ITSO Enterprise wrote a stand-alone Java application to generate an audit report from the archived data.

Example 6-72 illustrates a stand-alone Java program called FTEAudit.java to prepare the auditing data.

Example 6-72 FTEAudit.java for querying the database FTELOG

```
import java.sql.*;

public class FTEAudit {

    public static void main(String argv[])
    {

        String serverName    = "fileadmin.itso.com";
        String serverPort    = "50000";
        String databaseName  = "FTELOG";
        String userID        = "wmquser";
        String password      = "*****";
        String query         = null;
        String url           = null;
        Connection con       = null;
        int counter          = 0;

        try {
            Class.forName("com.ibm.db2.jcc.DB2Driver").newInstance();
            url = "jdbc:db2://" + serverName + ":" + serverPort + "/" + databaseName;
            con = DriverManager.getConnection(url,userID,password);
            query = "SELECT T.TRANSFER_ID,T.START_ID,T.RESULTTEXT,T.STATUS," +
                "S.ACTION_TIME,S.SOURCE_AGENT,S.SOURCE_QM,S.DESTINATION_AGENT," +
                "S.DESTINATION_QM,R.FILE_MODE,R.SOURCE_FILENAME,R.DESTINATION_FILENAME"+
                "FROM FTELOG.TRANSFER T JOIN FTELOG.TRANSFER_EVENT S ON " +
                "T.START_ID = S.ID JOIN FTELOG.TRANSFER_ITEM R ON " +
                "T.TRANSFER_ID=R.TRANSFER_ID WHERE S.ACTION_TIME>='"+argv[0]+
                "' AND S.ACTION_TIME<='"+argv[1]+"''";

            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery(query);
            System.out.println("*****");
            System.out.println("***** ITSO FTE AUDIT UTILITY *****");
            System.out.println("*****");
            System.out.println("Reporting Period :");
            System.out.println("Start Date (yyyy-mm-dd) = "+argv[0]);
            System.out.println("End   Date (yyyy-mm-dd) = "+argv[1]);
            System.out.println("*****\n");

            while (rs.next())
            {
                System.out.println("File Transfer Result = "+rs.getRow());
                System.out.println(
                    "TRANSFER_ID      = " + rs.getString(1)  + "\n"+
                    "START_ID          = " + rs.getString(2)  + "\n"+
                    "RESULTTEXT       = " + rs.getString(3)  + "\n"+
                    "STATUS           = " + rs.getString(4)  + "\n"+

```

```

        "ACTION_TIME           = " + rs.getString(5) + "\n"+
        "SOURCE_AGENT          = " + rs.getString(6) + "\n"+
        "SOURCE_QM              = " + rs.getString(7) + "\n"+
        "DESTINATION_AGENT      = " + rs.getString(8) + "\n"+
        "DESTINATION_QM         = " + rs.getString(9) + "\n"+
        "FILE_MODE              = " + rs.getString(10) + "\n"+
        "SOURCE_FILENAME         = " + rs.getString(11) + "\n"+
        "DESTINATION_FILENAME    = " + rs.getString(12) + "\n";
        counter++;
    }
    System.out.println("*****");
    System.out.println("Total Files Transferred = "+counter);
    System.out.println("*****");
    } catch (IllegalAccessException e) {
        e.printStackTrace();
    } catch (InstantiationException e) {
        e.printStackTrace();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
    } try {
        con.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
    }
}
}
}

```

FTEAudit.java is a stand-alone Java client, which uses the DB2 Type 4 JDBC driver to connect to the database FTELOG. This program takes two inputs, start date (*yyyy-MM-dd*) and end date (*yyyy-MM-dd*), and displays all the files that were transferred between those dates.

Example 6-73 illustrates the syntax for invoking the instance of the stand-alone program FTEAudit.java.

Example 6-73 Syntax to run FTEAudit.java

```
java FTEAudit 2011-07-10 2011-07-11
```

Example 6-74 illustrates the report that was produced by executing the audit program FTEAudit.java.

Example 6-74 Output from the execution of program FTEAudit.java

```

*****
***** ITSO FTE AUDIT UTILITY *****
*****
Reporting Period :
Start Date (yyyy-mm-dd) = 2011-07-10
End   Date (yyyy-mm-dd) = 2011-07-11
*****

```

File Transfer Result = 1
TRANSFER_ID = 414d51204954534f2e46494c452e434f1a20194e20001f0f
START_ID = 540
RESULTTEXT = BFGRP0032I: The file transfer request has successfully completed.
STATUS = success
ACTION_TIME = 2011-07-10 03:50:54.906
SOURCE_AGENT = STORE.CHICAGO.QM
SOURCE_QM = STORE.CHICAGO.QM
DESTINATION_AGENT = WAREHOUSE.AGENT.QM
DESTINATION_QM = WAREHOUSE.AGENT.QM
FILE_MODE = binary
SOURCE_FILENAME = E:\ITSO.INVENTORY.FILES\CHICAGO\ITSOCHI.INVENTORY.csv
DESTINATION_FILENAME = E:\ITSO.INVENTORY.FILES\WAREHOUSE\ITSOCHI.INVENTORY.csv

File Transfer Result = 2
TRANSFER_ID = 414d51204954534f2e46494c452e434f1a20194e20001f13
START_ID = 542
RESULTTEXT = BFGRP0032I: The file transfer request has successfully completed.
STATUS = success
ACTION_TIME = 2011-07-10 03:50:55.593
SOURCE_AGENT = STORE.CHICAGO.QM
SOURCE_QM = STORE.CHICAGO.QM
DESTINATION_AGENT = WAREHOUSE.AGENT.QM
DESTINATION_QM = WAREHOUSE.AGENT.QM
FILE_MODE = binary
SOURCE_FILENAME = E:\ITSO.INVENTORY.FILES\CHICAGO\ITSOCHI.INVENTORY.go
DESTINATION_FILENAME = E:\ITSO.INVENTORY.FILES\WAREHOUSE\ITSOCHI.INVENTORY.go

File Transfer Result = 3
TRANSFER_ID = 414d51204954534f2e46494c452e434f1a20194e2000240d
START_ID = 544
RESULTTEXT = BFGRP0032I: The file transfer request has successfully completed.
STATUS = success
ACTION_TIME = 2011-07-10 03:51:45.734
SOURCE_AGENT = STORE.ATLANTA.QM
SOURCE_QM = STORE.ATLANTA.QM
DESTINATION_AGENT = WAREHOUSE.AGENT.QM
DESTINATION_QM = WAREHOUSE.AGENT.QM
FILE_MODE = binary
SOURCE_FILENAME = E:\ITSO.INVENTORY.FILES\ATLANTA\ITSOATL.INVENTORY.csv
DESTINATION_FILENAME = E:\ITSO.INVENTORY.FILES\WAREHOUSE\ITSOATL.INVENTORY.csv

File Transfer Result = 4
TRANSFER_ID = 414d51204954534f2e46494c452e434f1a20194e20002411
START_ID = 546
RESULTTEXT = BFGRP0032I: The file transfer request has successfully completed.
STATUS = success
ACTION_TIME = 2011-07-10 03:51:46.187
SOURCE_AGENT = STORE.ATLANTA.QM
SOURCE_QM = STORE.ATLANTA.QM
DESTINATION_AGENT = WAREHOUSE.AGENT.QM
DESTINATION_QM = WAREHOUSE.AGENT.QM
FILE_MODE = binary
SOURCE_FILENAME = E:\ITSO.INVENTORY.FILES\ATLANTA\ITSOATL.INVENTORY.go
DESTINATION_FILENAME = E:\ITSO.INVENTORY.FILES\WAREHOUSE\ITSOATL.INVENTORY.go

```

File Transfer Result = 5
TRANSFER_ID         = 414d51204954534f2e46494c452e434f1a20194e2000290d
START_ID            = 548
RESULTTEXT         = BFGRP0032I: The file transfer request has successfully completed.
STATUS             = success
ACTION_TIME        = 2011-07-10 03:52:38.421
SOURCE_AGENT       = STORE.NEWYORK.QM
SOURCE_QM          = STORE.NEWYORK.QM
DESTINATION_AGENT  = WAREHOUSE.AGENT.QM
DESTINATION_QM     = WAREHOUSE.AGENT.QM
FILE_MODE          = binary
SOURCE_FILENAME    = E:\ITSO.INVENTORY.FILES\NEWYORK\ITSONEW.INVENTORY.csv
DESTINATION_FILENAME = E:\ITSO.INVENTORY.FILES\WAREHOUSE\ITSONEW.INVENTORY.csv

```

```

File Transfer Result = 6
TRANSFER_ID         = 414d51204954534f2e46494c452e434f1a20194e20002911
START_ID            = 550
RESULTTEXT         = BFGRP0032I: The file transfer request has successfully completed.
STATUS             = success
ACTION_TIME        = 2011-07-10 03:52:39.031
SOURCE_AGENT       = STORE.NEWYORK.QM
SOURCE_QM          = STORE.NEWYORK.QM
DESTINATION_AGENT  = WAREHOUSE.AGENT.QM
DESTINATION_QM     = WAREHOUSE.AGENT.QM
FILE_MODE          = binary
SOURCE_FILENAME    = E:\ITSO.INVENTORY.FILES\NEWYORK\ITSONEW.INVENTORY.go
DESTINATION_FILENAME = E:\ITSO.INVENTORY.FILES\WAREHOUSE\ITSONEW.INVENTORY.go

```

```

*****
Total Files Transferred = 6
*****

```

FTEAudit.java: The FTEAudit.java program queries data from the TRANSFER, TRANSFER_EVENT, and TRANSFER_ITEM tables using SQL joins. You can modify the query based on your functional requirements.

After this report was presented to ITSO Enterprise, its staff was able to prepare weekly audit reports for the inventory files that were transferred between the ITSO Enterprise stores and the ITSO Enterprise warehouse.

6.8 Business scenario 4

In this business scenario, we demonstrate the centralized administration feature of WebSphere MQ File Transfer Edition using MQ Explorer.

6.8.1 Requirement

ITSO Enterprise requires graphical administration of the following tasks:

- ▶ Monitor the availability states of the File Transfer Edition agent
- ▶ Monitor real-time file transfer progress

6.8.2 Solution

WebSphere MQ File Transfer Edition provides additional graphical administration and monitoring tool as an MQ Explorer plug-in. Install this tool through the Remote Tools CD.

The prerequisites to run the WebSphere MQ Explorer plug-in are any platform that is supported for the MQ Explorer by WebSphere MQ. Check the “Detailed system requirements” page for the required platform at this website:

<http://www.ibm.com/software/integration/wmq/requirements/#WebSphereMQV70/701>

During the installation, choose this component in the “Select Set of Features to install” window or click **Complete installation**. Figure 6-36 on page 289 illustrates the various installation options.

To launch the MQ Explorer Managed File Transfer plug-in for WebSphere MQ File Transfer Edition, follow the steps:

1. On the server fileadmin.itso.com, select **Start Programs** → **IBM WebSphere MQ** → **WebSphere MQ Explorer**, and click **WebSphere MQ Explorer** to launch the Eclipse GUI.
2. In the Navigator view of WebSphere MQ Explorer, expand **Managed File Transfer** → **ITSO.FILE.COORDINATION.QM**.

Figure 6-40 illustrates the various monitoring and administration options of this plug-in.

If the WebSphere MQ Explorer plug-in is not displayed in your WebSphere MQ Explorer automatically after installation, see the resolution steps at the following website:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.doc/install_all_mq_explorer.htm

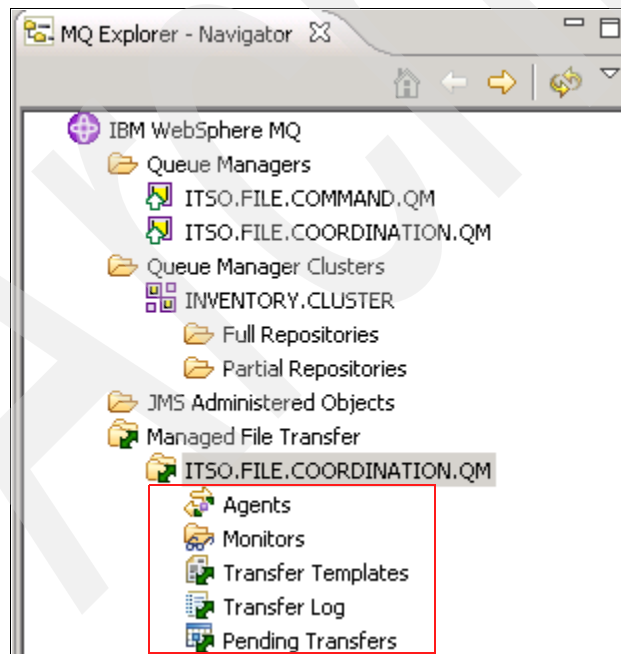


Figure 6-40 MQ Explorer plug-in for the File Transfer Edition

6.8.3 Testing centralized administration

Now that the MQ Explorer plug-in is installed, we can test the requirements that are requested by ITSO Enterprise.

Monitoring the availability of the File Transfer Edition agents

To test this functional requirement, double-click **Agents**, as illustrated in Figure 6-40 on page 297. This option opens an MQ Explorer - Content view on the right, which has details about all the File Transfer Edition agents that are registered with the coordination queue manager ITSO.FILE.COORDINATION.QM. Figure 6-41 illustrates the MQ Explorer - Content view with the File Transfer Edition agent status and details.

Name	Description	Status	Source transfer	Destination
STORE.ATLANTA.QM	Agent for ITSO Atlanta Store	Ready	0	0
STORE.CHICAGO.QM	Agent for ITSO Chicago Store	Ready	0	0
STORE.NEWYORK.QM	Agent for ITSO NewYork Store	Ready	0	0
WAREHOUSE.AGENT.QM	Agent for ITSO Warehouse	Ready	0	0

Figure 6-41 File Transfer Edition agent details in WebSphere MQ Explorer

Several values are possible for the status of the File Transfer Edition agent. Table 6-4 illustrates the various states of the File Transfer Edition agents.

Table 6-4 File Transfer Edition agent status values

Status	Explanation
ACTIVE	The agent is running and sending or receiving files. The agent publishes its status at regular intervals. The last update was received within the expected time period.
READY	The agent is running, but it is not sending or receiving files. The agent publishes its status at regular intervals. The last update was received within the expected time period.
STARTING	The agent started, but it is not yet ready to perform transfers.
UNREACHABLE	Agent status updates were not received at the expected time intervals. The agent might have stopped running due to an error, it might have been shut down abruptly, or it might be running but experiencing communication problems.
STOPPED	The agent has been stopped. It was shut down in a controlled manner.
NO_INFORMATION	The agent version might be WebSphere MQ File Transfer Edition Version 7.0.2 or earlier. The agent is not publishing updates in a form that this command can process.
UNKNOWN	The status of the agent cannot be determined. It might have published a status that is not recognized by this tool. If you have mixed product versions on your network, upgrading the installation version of this tool might fix this problem.

Status	Explanation
PROBLEM	The agent command handler might not work. The agent publishes status messages, but these status messages are out-of-date.

To simulate a change in the state of the File Transfer Edition agent values, we stop the agent on the ITSO Enterprise warehouse server by using the command that is illustrated in Example 6-75.

Example 6-75 Stop the File Transfer Edition agent

```
fteStopAgent WAREHOUSE.AGENT.QM
```

Figure 6-42 illustrates the Stopped status of the ITSO Enterprise warehouse agent.

Name	Description	Status	Source transfer	Destination
STORE.ATLANTA.QM	Agent for ITSO Atlanta Store	Ready	0	0
STORE.CHICAGO.QM	Agent for ITSO Chicago Store	Ready	0	0
STORE.NEWYORK.QM	Agent for ITSO NewYork Store	Ready	0	0
WAREHOUSE.AGENT.QM	Agent for ITSO Warehouse	Stopped	0	0

Figure 6-42 ITSO Enterprise warehouse agent status is Stopped

You can also test the connectivity of the File Transfer Edition agent from this view. Right-click the agent name and select **Test Connectivity**. Figure 6-43 illustrates the option of testing the agent connectivity.

Name	Description	Status	Source transfer	Destination
STORE.ATLANTA.QM	Agent for ITSO Atlanta Store	Ready	0	0
STORE.CHICAGO.QM	Agent for IT	Ready	0	0
STORE.NEWYORK.QM	Agent for IT	Ready	0	0
WAREHOUSE.AGENT.QM	Agent for ITSO warehouse	Ready	0	0

Figure 6-43 Agent connectivity test from MQ Explorer

Clicking **Test Connectivity** opens a window. Click **Test** for checking the connectivity of the File Transfer Edition agent. Figure 6-44 on page 300 illustrates the connectivity result for the ITSO Enterprise agent STORE.ATLANTA.QM.

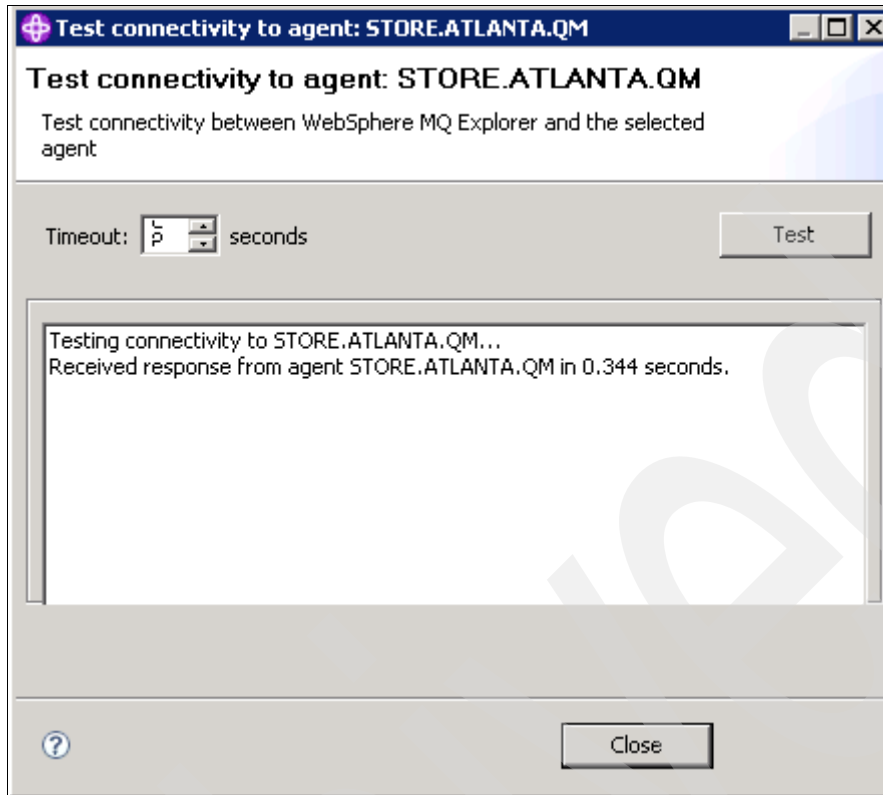


Figure 6-44 Connectivity test for ITSO Enterprise Store agent STORE.ATLANTA.QM

Monitoring real-time file transfer progress

You can monitor a file transfer that is in progress by using the Managed File Transfer - Current Transfer Progress tab in WebSphere MQ Explorer. You can start this file transfer from either WebSphere MQ Explorer or the command line. The tab also displays the progress of the scheduled transfers at the point that the scheduled transfers start.

Figure 6-45 illustrates an example of the file transfer progress from the ITSO Enterprise Chicago store to the ITSO Enterprise warehouse.

Current File	File Number	Progress	Rate	Started (America/New_York)
CHICAGO.INVENTORY.csv - (760MiB / 2GiB)	1 / 1	<div style="width: 25%; background-color: blue; height: 10px;"></div> 25%	29MiB/s	6/27/11 5:03:56 PM EDT

Figure 6-45 File transfer progress

Previous file transfer information is not retained after you stop and restart WebSphere MQ Explorer. At restart, the information about past transfers is cleared from the Current Transfer Progress tab. After you have started a new file transfer using WebSphere MQ Explorer or the command line, you can monitor the progress of your transfer in the Current Transfer Progress tab. The following information is displayed for each transfer in progress:

- ▶ **Source:** The name of the agent that is used to transfer the file from the source system.
- ▶ **Destination:** The name of the agent that is used to receive the file at the destination system.

- ▶ **Current file:** The name of the file currently being transferred. The part of the individual file that has already been transferred is displayed in B, KiB, MiB, GiB, or TiB along with the total size of the file in parentheses. The unit of measurement displayed depends on the size of the file. B is bytes per second. KiB/s is kibibytes per second, where 1 kibibyte equals 1024 bytes. MiB/s is mebibytes per second, where 1 mebibyte equals 1 048 576 bytes. GiB/s is gibibytes per second where 1 gibibyte equals 1 073 741 824 bytes. TiB/s is tebibytes per second where 1 tebibyte equals 1 099 511 627 776 bytes.
- ▶ **File number:** If you are transferring more than one file, this number represents how far through the total group of files the transfer has gotten.
- ▶ **Progress:** The progress bar shows how complete the current file transfer is as a percentage.
- ▶ **Rate:** The rate that the file is being transferred in KiB/s (kibibytes per second, where 1 kibibyte equals 1024 bytes.)
- ▶ **Started (selected time zone):** The time that the transfer started in the administrator's selected time zone.

Refresh: This tab regularly refreshes its information automatically, but to force a refreshed view of what is displayed in the Current Transfer Progress tab, click the **Refresh** icon on the Content view toolbar.

With these two functional requirements implemented for ITSO Enterprise, its staff can easily monitor the states of agents and the real-time file transfer progress. The WebSphere MQ File Transfer Edition plug-in for MQ Explorer helped ITSO Enterprise to centrally administer the runtime components and file transfers in progress.

6.9 Business scenario 5

In this business scenario, we demonstrate how to configure the secure sockets layer (SSL) connection from the WebSphere MQ File Transfer Edition agent to the coordination queue manager, command queue manager, and agent queue manager.

6.9.1 Requirement

ITSO Enterprise has following requirements:

- ▶ All communication between the File Transfer Edition agents and ITSO Enterprise queue managers must be encrypted and secured.
- ▶ The administration of the File Transfer Edition must only be performed by users that are authorized to administer the File Transfer Edition.

6.9.2 Solution

This business scenario shows how to configure two-way authentication between the File Transfer Edition agents to the command queue manager, coordination queue manager, and agent queue manager.

Follow these steps to implement this business solution:

1. Create the agent truststore
2. Create the queue manager certificate store
3. Create and extract the agent self-signed certificate

4. Create and extract the queue manager self-signed certificate
5. Add the queue manager certificate to the agent store
6. Add the agent certificate to the queue manager store
7. Configure the queue manager channel for secure communication
8. Configure the agent, command, and coordination properties

Creating the agent truststore

We create a Java truststore and keystore for the ITSO Enterprise Atlanta store agent, ITSO Enterprise Chicago store agent, ITSO Enterprise NEWYORK store agent, and ITSO Enterprise warehouse agent.

Example 6-76 illustrates the commands for creating a Java truststore and keystore for the ITSO Enterprise Atlanta store on the server store.atlanta.itso.com.

Example 6-76 Create truststore and keystore for Atlanta File Transfer Edition agent

```
runmqckm -keydb -create -db E:\SSL\ATLANTA\truststore.jks -pw itso4you -type JKS
runmqckm -keydb -create -db E:\SSL\ATLANTA\keystore.jks -pw itso4you -type JKS
```

Example 6-77 illustrates the commands for creating a Java truststore and keystore for the ITSO Enterprise Chicago store on the server store.chicago.itso.com.

Example 6-77 Create truststore and keystore for Chicago File Transfer Edition agent

```
runmqckm -keydb -create -db E:\SSL\CHICAGO\truststore.jks -pw itso4you -type JKS
runmqckm -keydb -create -db E:\SSL\CHICAGO\keystore.jks -pw itso4you -type JKS
```

Example 6-78 illustrates the commands for creating a Java truststore and keystore for the ITSO Enterprise NEWYORK store on the server store.newyork.itso.com.

Example 6-78 Create truststore and keystore for the New York File Transfer Edition agent

```
runmqckm -keydb -create -db E:\SSL\NEWYORK\truststore.jks -pw itso4you -type JKS
runmqckm -keydb -create -db E:\SSL\NEWYORK\keystore.jks -pw itso4you -type JKS
```

Example 6-79 illustrates the commands for creating a Java truststore and keystore for the ITSO Enterprise warehouse on the server fileadmin.itso.com.

Example 6-79 Create truststore and keystore for warehouse File Transfer Edition agent

```
runmqckm -keydb -create -db E:\SSL\WAREHOUSE\truststore.jks -pw itso4you -type JKS
runmqckm -keydb -create -db E:\SSL\WAREHOUSE\keystore.jks -pw itso4you -type JKS
```

Example 6-80 illustrates the commands for creating a Java truststore and keystore for MQ Explorer and the administration utilities on the server fileadmin.itso.com.

Example 6-80 Command to create truststore and keystore for MQ Explorer

```
runmqckm -keydb -create -db E:\SSL\EXPLORER\truststore.jks -pw itso4you -type JKS
runmqckm -keydb -create -db E:\SSL\EXPLORER\keystore.jks -pw itso4you -type JKS
```

MQ Explorer: The MQ Explorer truststore and keystore are used for secure administration from the MQ Explorer and command-line utilities.

Creating the queue manager certificate store

We create the Certificate Management System (CMS) keystore for the ITSO Enterprise stores and ITSO Enterprise warehouse queue manager.

Example 6-81 illustrates the command for creating the CMS keystore key.kdb for the Atlanta store queue manager STORE.ATLANTA.QM on the server store.atlanta.itso.com.

Example 6-81 Command to create the STORE.ATLANTA.QM keystore

```
runmqckm -keydb -create -db C:\WMQ\Qmgrs\STORE!ATLANTA!QM\ssl\key.kdb -pw itso4you -type cms -expire 365 -stash
```

Example 6-82 illustrates the command for creating the CMS keystore key.kdb for the Chicago store queue manager STORE.CHICAGO.QM on the server store.chicago.itso.com.

Example 6-82 Command to create the STORE.CHICAGO.QM keystore

```
runmqckm -keydb -create -db C:\WMQ\Qmgrs\STORE!CHICAGO!QM\ssl\key.kdb -pw itso4you -type cms -expire 365 -stash
```

Example 6-83 illustrates the command for creating the CMS keystore key.kdb for the New York store queue manager STORE.NEWYORK.QM on the server store.newyork.itso.com.

Example 6-83 Command to create the STORE.NEWYORK.QM keystore

```
runmqckm -keydb -create -db C:\WMQ\Qmgrs\STORE!NEWYORK!QM\ssl\key.kdb -pw itso4you -type cms -expire 365 -stash
```

Example 6-84 illustrates the command for creating the CMS keystore key.kdb for the warehouse queue manager WAREHOUSE.AGENT.QM on the server warehouse.itso.com.

Example 6-84 Command to create the WAREHOUSE.AGENT.QM keystore

```
runmqckm -keydb -create -db C:\WMQ\Qmgrs\WAREHOUSE!AGENT!QM\ssl\key.kdb -pw itso4you -type cms -expire 365 -stash
```

Example 6-85 illustrates the command for creating the CMS keystore key.kdb for the coordination queue manager ITSO.FILE.COORDINATION.QM on the server fileadmin.itso.com.

Example 6-85 Command to create the ITSO.FILE.COORDINATION.QM keystore

```
runmqckm -keydb -create -db C:\WMQ\Qmgrs\ITSO!FILE!COORDINATION!QM\ssl\key.kdb -pw itso4you -type cms -expire 365 -stash
```

Example 6-86 illustrates the command for creating the CMS keystore key.kdb for the command queue manager ITSO.FILE.COMMAND.QM on the server fileadmin.itso.com.

Example 6-86 Command to create the ITSO.FILE.COMMAND.QM keystore

```
runmqckm -keydb -create -db C:\WMQ\Qmgrs\ITSO!FILE!COMMAND!QM\ssl\key.kdb -pw itso4you -type cms -expire 365 -stash
```

Creating and extracting the agent self-signed certificates

We create and extract the self-signed certificates from the agent keystores.

Example 6-87 illustrates the commands to create and extract the certificates from the ITSO Enterprise Atlanta store agent keystore on the server store.atlanta.itso.com.

Example 6-87 Create and extract the certificate from the Atlanta store agent keystore

```
runmqckm -cert -create -db E:\SSL\ATLANTA\keystore.jks -pw itso4you -label  
atlantaagent -dn CN=itso -size 1024 -x509version 3 -expire 365
```

```
runmqckm -cert -extract -db E:\SSL\ATLANTA\keystore.jks -pw itso4you -label  
atlantaagent -target E:\SSL\ATLANTA\atlantaagent.der -format binary
```

Example 6-88 illustrates the commands to create and extract the certificates from the ITSO Enterprise Chicago store agent keystore on the server store.chicago.itso.com.

Example 6-88 Create and extract the certificate from the Chicago store agent keystore

```
runmqckm -cert -create -db E:\SSL\CHICAGO\keystore.jks -pw itso4you -label  
chicagoagent -dn CN=itso -size 1024 -x509version 3 -expire 365
```

```
runmqckm -cert -extract -db E:\SSL\CHICAGO\keystore.jks -pw itso4you -label  
chicagoagent -target E:\SSL\CHICAGO\chicagoagent.der -format binary
```

Example 6-89 illustrates the commands to create and extract the certificates from the ITSO Enterprise New York store agent keystore on the server store.newyork.itso.com.

Example 6-89 Create and extract the certificates from the NEWYORK store agent keystore

```
runmqckm -cert -create -db E:\SSL\NEWYORK\keystore.jks -pw itso4you -label  
newyorkagent -dn CN=itso -size 1024 -x509version 3 -expire 365
```

```
runmqckm -cert -extract -db E:\SSL\NEWYORK\keystore.jks -pw itso4you -label  
newyorkagent -target E:\SSL\NEWYORK\newyorkagent.der -format binary
```

Example 6-90 illustrates the commands to create and extract the certificates from the ITSO Enterprise warehouse agent keystore on the server warehouse.itso.com.

Example 6-90 Create and extract the certificates from the warehouse agent keystore

```
runmqckm -cert -create -db E:\SSL\WAREHOUSE\keystore.jks -pw itso4you -label  
warehouseagent -dn CN=itso -size 1024 -x509version 3 -expire 365
```

```
runmqckm -cert -extract -db E:\SSL\WAREHOUSE\keystore.jks -pw itso4you -label  
warehouseagent -target E:\SSL\WAREHOUSE\warehouseagent.der -format binary
```

Example 6-91 illustrates the commands to create and extract the certificates from the MQ Explorer keystore on the server fileadmin.itso.com.

Example 6-91 Create and extract certificates from the MQ Explorer keystore

```
runmqckm -cert -create -db E:\SSL\EXPLORER\keystore.jks -pw itso4you -label  
explorer -dn CN=itso -size 1024 -x509version 3 -expire 365
```

```
runmqckm -cert -extract -db E:\SSL\EXPLORER\keystore.jks -pw itso4you -label  
explorer -target E:\SSL\EXPLORER\explorer.der -format binary
```

Creating and extracting the queue manager self-signed certificates

We create and extract the self-signed certificates from the store, warehouse, command, and coordination queue manager keystores.

Example 6-92 illustrates the commands to create and extract the certificate from the queue manager STORE.ATLANTA.QM keystore on the server store.atlanta.itso.com.

Example 6-92 Create and extract the certificate from the STORE.ATLANTA.QM's keystore

```
runmqckm -cert -create -db C:\WMQ\Qmgrs\STORE!ATLANTA!QM\ssl\key.kdb -pw itso4you  
-label ibmwebspheremqstore.atlanta.qm -dn CN=itso -size 1024 -x509version 3  
-expire 365
```

```
runmqckm -cert -extract -db C:\WMQ\Qmgrs\STORE!ATLANTA!QM\ssl\key.kdb -pw itso4you  
-label ibmwebspheremqstore.atlanta.qm -target  
E:\SSL\ATLANTA\ibmwebspheremqstore.atlanta.qm.der -format binary
```

Example 6-93 illustrates the commands to create and extract the certificate from the queue manager STORE.CHICAGO.QM keystore on the server store.chicago.itso.com.

Example 6-93 Create and extract the certificate from the STORE.CHICAGO.QM's keystore

```
runmqckm -cert -create -db C:\WMQ\Qmgrs\STORE!CHICAGO!QM\ssl\key.kdb -pw itso4you  
-label ibmwebspheremqstore.chicago.qm -dn CN=itso -size 1024 -x509version 3  
-expire 365
```

```
runmqckm -cert -extract -db C:\WMQ\Qmgrs\STORE!CHICAGO!QM\ssl\key.kdb -pw itso4you  
-label ibmwebspheremqstore.chicago.qm -target  
E:\SSL\CHICAGO\ibmwebspheremqstore.chicago.qm.der -format binary
```

Example 6-94 illustrates the commands to create and extract the certificate from the queue manager STORE.NEWYORK.QM keystore on the server store.newyork.itso.com.

Example 6-94 Create and extract the certificate from the STORE.NEWYORK.QM's keystore

```
runmqckm -cert -create -db C:\WMQ\Qmgrs\STORE!NEWYORK!QM\ssl\key.kdb -pw itso4you  
-label ibmwebspheremqstore.newyork.qm -dn CN=itso -size 1024 -x509version 3  
-expire 365
```

```
runmqckm -cert -extract -db C:\WMQ\Qmgrs\STORE!NEWYORK!QM\ssl\key.kdb -pw itso4you  
-label ibmwebspheremqstore.newyork.qm -target  
E:\SSL\NEWYORK\ibmwebspheremqstore.newyork.qm.der -format binary
```

Example 6-95 illustrates the commands to create and extract the certificate from the queue manager WAREHOUSE.AGENT.QM keystore on the server warehouse.itso.com.

Example 6-95 Create and extract certificate from WAREHOUSE.AGENT.QM's keystore

```
runmqckm -cert -create -db C:\WMQ\Qmgrs\WAREHOUSE!AGENT!QM\ssl\key.kdb -pw  
itso4you -label ibmwebspheremqwarehouse.agent.qm -dn CN=itso -size 1024  
-x509version 3 -expire 365
```

```
runmqckm -cert -extract -db C:\WMQ\Qmgrs\WAREHOUSE!AGENT!QM\ssl\key.kdb -pw  
itso4you -label ibmwebspheremqwarehouse.agent.qm -target  
E:\SSL\WAREHOUSE\ibmwebspheremqwarehouse.agent.qm.der -format binary
```

Example 6-96 illustrates the commands to create and extract the certificate from the queue manager ITSO.FILE.COMMAND.QM keystore on the server fileadmin.itso.com.

Example 6-96 Create and extract certificate from ITSO.FILE.COMMAND.QM's keystore

```
runmqckm -cert -create -db C:\WMQ\Qmgrs\ITSO!FILE!COMMAND!QM\ssl\key.kdb -pw  
itso4you -label ibmwebspheremqitso.file.command.qm -dn CN=itso -size 1024  
-x509version 3 -expire 365
```

```
runmqckm -cert -extract -db C:\WMQ\Qmgrs\ITSO!FILE!COMMAND!QM\ssl\key.kdb -pw  
itso4you -label ibmwebspheremqitso.file.command.qm -target  
E:\SSL\COMMAND\ibmwebspheremqitso.file.command.qm.der -format binary
```

Example 6-97 illustrates the commands to create and extract the certificate from the queue manager ITSO.FILE.COORDINATION.QM on the server fileadmin.itso.com.

Example 6-97 Create and extract certificate from ITSO.FILE.COORDINATION.QM's keystore

```
runmqckm -cert -create -db C:\WMQ\Qmgrs\ITSO!FILE!COORDINATION!QM\ssl\key.kdb -pw  
itso4you -label ibmwebspheremqitso.file.coordination.qm -dn CN=itso -size 1024  
-x509version 3 -expire 365
```

```
runmqckm -cert -extract -db C:\WMQ\Qmgrs\ITSO!FILE!COORDINATION!QM\ssl\key.kdb -pw  
itso4you -label ibmwebspheremqitso.file.coordination.qm -target  
E:\SSL\COORDINATION\ibmwebspheremqitso.file.coordination.qm.der -format binary
```

Adding the queue manager certificate to the FTE agent truststore

We import the extracted queue manager certificates in the agent truststores.

Names: Copy or FTP command the queue manager certificate `ibmwebspheremqitso.file.command.qm.der` and coordination queue manager certificate `ibmwebspheremqitso.file.coordination.qm.der` from the server `fileadmin.itso.com` to the servers `store.atlanta.itso.com`, `store.chicago.itso.com`, `store.newyork.itso.com`, and `warehouse.itso.com`.

Example 6-98 illustrates the commands to import the certificate from the ITSO Enterprise Atlanta store queue manager, command queue manager, and coordination queue manager in the Atlanta File Transfer Edition agent truststore.

Example 6-98 Import certificates in the Atlanta agent truststore

```
runmqckm -cert -add -db E:\SSL\ATLANTA\truststore.jks -pw itso4you -type JKS  
-label ibmwebspheremqstore.atlanta.qm -file  
E:\SSL\ATLANTA\ibmwebspheremqstore.atlanta.qm.der -format binary -trust enable
```

```
runmqckm -cert -add -db E:\SSL\ATLANTA\truststore.jks -pw itso4you -type JKS  
-label ibmwebspheremqitso.file.command.qm -file  
E:\SSL\ATLANTA\ibmwebspheremqitso.file.command.qm.der -format binary -trust enable
```

```
runmqckm -cert -add -db E:\SSL\ATLANTA\truststore.jks -pw itso4you -type JKS  
-label ibmwebspheremqitso.file.coordination.qm -file  
E:\SSL\ATLANTA\ibmwebspheremqitso.file.coordination.qm.der -format binary -trust  
enable
```

Example 6-99 illustrates the commands to import the certificates from the ITSO Enterprise Chicago store queue manager, command queue manager, and coordination queue manager in the ITSO Enterprise Chicago File Transfer Edition agent truststore.

Example 6-99 Import certificates in the Chicago agent truststore

```
runmqckm -cert -add -db E:\SSL\CHICAGO\truststore.jks -pw itso4you -type JKS  
-label ibmwebspheremqstore.chicago.qm -file  
E:\SSL\CHICAGO\ibmwebspheremqstore.chicago.qm.der -format binary -trust enable
```

```
runmqckm -cert -add -db E:\SSL\CHICAGO\truststore.jks -pw itso4you -type JKS  
-label ibmwebspheremqitso.file.command.qm -file  
E:\SSL\CHICAGO\ibmwebspheremqitso.file.command.qm.der -format binary -trust enable
```

```
runmqckm -cert -add -db E:\SSL\CHICAGO\truststore.jks -pw itso4you -type JKS  
-label ibmwebspheremqitso.file.coordination.qm -file  
E:\SSL\CHICAGO\ibmwebspheremqitso.file.coordination.qm.der -format binary -trust  
enable
```

Example 6-100 illustrates the commands to import the certificates from the ITSO Enterprise NEWYORK store queue manager, command queue manager, and coordination queue manager in the ITSO Enterprise NEWYORK File Transfer Edition agent truststore.

Example 6-100 Import certificates in the NEWYORK agent truststore

```
runmqckm -cert -add -db E:\SSL\NEWYORK\truststore.jks -pw itso4you -type JKS  
-label ibmwebspheremqstore.newyork.qm -file  
E:\SSL\NEWYORK\ibmwebspheremqstore.newyork.qm.der -format binary -trust enable
```

```
runmqckm -cert -add -db E:\SSL\NEWYORK\truststore.jks -pw itso4you -type JKS  
-label ibmwebspheremqitso.file.command.qm -file  
E:\SSL\NEWYORK\ibmwebspheremqitso.file.command.qm.der -format binary -trust enable
```

```
runmqckm -cert -add -db E:\SSL\NEWYORK\truststore.jks -pw itso4you -type JKS  
-label ibmwebspheremqitso.file.coordination.qm -file  
E:\SSL\NEWYORK\ibmwebspheremqitso.file.coordination.qm.der -format binary -trust  
enable
```

Example 6-101 illustrate commands to import certificates from the ITSO Enterprise warehouse queue manager, command queue manager, and coordination queue manager in the ITSO Enterprise warehouse File Transfer Edition agent truststore.

Example 6-101 Import certificates in the warehouse agent truststore

```
runmqckm -cert -add -db E:\SSL\WAREHOUSE\truststore.jks -pw itso4you -type JKS  
-label ibmwebspheremqwarehouse.agent.qm -file  
E:\SSL\WAREHOUSE\ibmwebspheremqwarehouse.agent.qm.der -format binary -trust enable
```

```
runmqckm -cert -add -db E:\SSL\WAREHOUSE\truststore.jks -pw itso4you -type JKS  
-label ibmwebspheremqitso.file.command.qm -file  
E:\SSL\WAREHOUSE\ibmwebspheremqitso.file.command.qm.der -format binary -trust  
enable
```

```
runmqckm -cert -add -db E:\SSL\WAREHOUSE\truststore.jks -pw itso4you -type JKS  
-label ibmwebspheremqitso.file.coordination.qm -file
```

```
E:\SSL\WAREHOUSE\ibmwebspheremqitso.file.coordination.qm.der -format binary -trust enable
```

Example 6-102 illustrates the commands to import the certificates from the command queue manager and coordination queue manager in the MQ Explorer truststore.

Example 6-102 Import certificate in the MQ Explorer agent truststore

```
runmqckm -cert -add -db E:\SSL\EXPLORER\truststore.jks -pw itso4you -type JKS  
-label ibmwebspheremqitso.file.command.qm -file  
E:\SSL\EXPLORER\ibmwebspheremqitso.file.command.qm.der -format binary -trust  
enable
```

```
runmqckm -cert -add -db E:\SSL\EXPLORER\truststore.jks -pw itso4you -type JKS  
-label ibmwebspheremqitso.file.coordination.qm -file  
E:\SSL\EXPLORER\ibmwebspheremqitso.file.coordination.qm.der -format binary -trust  
enable
```

Adding the agent certificate to the queue manager store

We add the extracted agent certificates inside the queue manager CMS keystore repository.

Example 6-103 illustrates the command to import the Atlanta File Transfer Edition agent certificate inside the queue manager STORE.ATLANTA.QM keystore repository.

Example 6-103 Import the certificate in the STORE.ATLANTA.QM keystore repository

```
runmqckm -cert -add -db C:\WMQ\Qmgrs\STORE!ATLANTA!QM\ssl\key.kdb -pw itso4you  
-type cms -label atlantaagent -file E:\SSL\ATLANTA\atlantaagent.der -format binary  
-trust enable
```

Example 6-104 illustrates the command to import the Chicago File Transfer Edition agent certificate inside the queue manager STORE.CHICAGO.QM keystore repository.

Example 6-104 Import the certificate in the STORE.NEWYORK.QM keystore repository

```
runmqckm -cert -add -db C:\WMQ\Qmgrs\STORE!CHICAGO!QM\ssl\key.kdb -pw itso4you  
-type cms -label chicagoagent -file E:\SSL\CHICAGO\chicagoagent.der -format binary  
-trust enable
```

Example 6-105 illustrates the command to import the NEWYORK File Transfer Edition agent certificate inside the queue manager STORE.NEWYORK.QM keystore repository.

Example 6-105 Import the certificate in the STORE.NEWYORK.QM keystore repository

```
runmqckm -cert -add -db C:\WMQ\Qmgrs\STORE!NEWYORK!QM\ssl\key.kdb -pw itso4you  
-type cms -label newyorkagent -file E:\SSL\NEWYORK\newyorkagent.der -format binary  
-trust enable
```

Example 6-106 on page 309 illustrates the command to import the warehouse File Transfer Edition agent certificate inside the queue manager WAREHOUSE.AGENT.QM keystore repository.

Example 6-106 Import the certificate in the WAREHOUSE.AGENT.QM keystore repository

```
runmqckm -cert -add -db C:\WMQ\Qmgrs\WAREHOUSE!AGENT!QM\ssl\key.kdb -pw itso4you  
-type cms -label warehouseagent -file E:\SSL\WAREHOUSE\warehouseagent.der -format  
binary -trust enable
```

Names: Copy or FTP the certificate atlantaagent.der from the server store.atlanta.itso.com, chicagoagent.der from the server store.chicago.itso.com, newyorkagent.der from the server store.itso.newyork.com, and the warehouseagent.der from warehouse.itso.com to the server fileadmin.itso.com.

Agents and administration utilities also communicate with the command queue manager and the coordination queue manager. Therefore, we add the certificates that were extracted from the agent and explorer keystores, into the command and coordination queue manager keystore repositories.

Example 6-107 illustrates the commands to import the certificates from the Atlanta, Chicago, NEWYORK, and warehouse File Transfer Edition agents and explorer certificates inside the queue manager ITSO.FILE.COMMAND.QM keystore repository.

Example 6-107 Import certificates in ITSO.FILE.COMMAND.QM keystore repository

```
runmqckm -cert -add -db C:\WMQ\Qmgrs\ITSO!FILE!COMMAND!QM\ssl\key.kdb -pw itso4you  
-type cms -label atlantaagent -file E:\SSL\atlantaagent.der -format binary -trust  
enable  
runmqckm -cert -add -db C:\WMQ\Qmgrs\ITSO!FILE!COMMAND!QM\ssl\key.kdb -pw itso4you  
-type cms -label chicagoagent -file E:\SSL\chicagoagent.der -format binary -trust  
enable  
runmqckm -cert -add -db C:\WMQ\Qmgrs\ITSO!FILE!COMMAND!QM\ssl\key.kdb -pw itso4you  
-type cms -label newyorkagent -file E:\SSL\newyorkagent.der -format binary -trust  
enable  
runmqckm -cert -add -db C:\WMQ\Qmgrs\ITSO!FILE!COMMAND!QM\ssl\key.kdb -pw itso4you  
-type cms -label warehouseagent -file E:\SSL\warehouseagent.der -format binary  
-trust enable  
runmqckm -cert -add -db C:\WMQ\Qmgrs\ITSO!FILE!COMMAND!QM\ssl\key.kdb -pw itso4you  
-type cms -label explorer -file E:\SSL\explorer.der -format binary -trust enable
```

Example 6-108 illustrates the commands to import the certificates from the Atlanta, Chicago, NEWYORK, and warehouse File Transfer Edition agents and explorer certificates inside the queue manager ITSO.FILE.COORDINATION.QM keystore repository.

Example 6-108 Import certificates in ITSO.FILE.COORDINATION.QM keystore repository

```
runmqckm -cert -add -db C:\WMQ\Qmgrs\ITSO!FILE!COORDINATION!QM\ssl\key.kdb -pw  
itso4you -type cms -label atlantaagent -file E:\SSL\atlantaagent.der -format  
binary -trust enable  
runmqckm -cert -add -db C:\WMQ\Qmgrs\ITSO!FILE!COORDINATION!QM\ssl\key.kdb -pw  
itso4you -type cms -label chicagoagent -file E:\SSL\chicagoagent.der -format  
binary -trust enable  
runmqckm -cert -add -db C:\WMQ\Qmgrs\ITSO!FILE!COORDINATION!QM\ssl\key.kdb -pw  
itso4you -type cms -label newyorkagent -file E:\SSL\newyorkagent.der -format  
binary -trust enable  
runmqckm -cert -add -db C:\WMQ\Qmgrs\ITSO!FILE!COORDINATION!QM\ssl\key.kdb -pw  
itso4you -type cms -label warehouseagent -file E:\SSL\warehouseagent.der -format  
binary -trust enable
```

```
runmqckm -cert -add -db C:\WMQ\Qmgrs\ITSO!FILE!COORDINATION!QM\ssl\key.kdb -pw  
itso4you -type cms -label explorer -file E:\SSL\explorer.der -format binary -trust  
enable
```

Configuring the queue manager channel for secure communication

File Transfer Edition agents, File Transfer Edition commands, and administration utilities connect to the agent, command, and coordination queue managers using the server connection channel. We created several connection channels ITSO.SSL.SVRCONN in “Creating the server connection channel” on page 254.

Now, we configure this channel to use two-way Secure Sockets Layer (SSL) authentication and the SSL cipher as RC4_MD5_US.

Example 6-109 illustrates the command, which must be executed on all the agent, command, and coordination queue managers. Execute this command from the **runmqsc** console of the queue manager STORE.ATLANTA.QM, STORE.CHICAGO.QM, STORE.NEWYORK.QM, WAREHOUSE.AGENT.QM, ITSO.FILE.COMMAND.QM, and ITSO.FILE.COORDINATION.QM.

Example 6-109 Alter server connection channel for SSL configurations

```
ALTER CHANNEL(ITSO.SSL.SVRCONN) CHLTYPE(SVRCONN) SSLCIPH(RC4_MD5_US)  
SSLCAUTH(REQUIRED)
```

Important: Stop the File Transfer Edition agents by using the **fteStopAgent** command before changing the SSL configuration on the server connection channels.

Configuring the agent, command, and coordination properties

In this section, we cover the agent, command, and coordination properties.

agent.properties file

To configure SSL on the File Transfer Edition agents, add the `agentSslCipherSpec`, `agentSslTrustStore`, `agentSslTrustStorePassword`, `agentSslKeyStore`, and `agentSslKeyStorePassword` properties in the `agent.properties` file of the Atlanta, Chicago, NEWYORK, and warehouse File Transfer Edition agents.

To get more details about the SSL properties for File Transfer Edition agents, see this website:

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.doc/props_ssl.htm

Figure 6-46 on page 311 illustrates the `agent.properties` file for the I TSO Atlanta store File Transfer Edition agent that has been updated with the SSL properties.

```
1 agent.properties
#
#Mon Jun 27 07:36:14 EDT 2011
agentQMgr=STORE.ATLANTA.QM
agentQMgrPort=1414
agentDesc=Agent for ITSO Atlanta Store
agentQMgrHost=store.atlanta.itso.com
agentQMgrChannel=ITSO.SSL.SVRCONN
agentName=STORE.ATLANTA.QM
commandPath=E:\\Scripts\\ATLANTA
agentSslCipherSpec=RC4_MD5_US
agentSslTrustStore=E:\\SSL\\ATLANTA\\truststore.jks
agentSslTrustStorePassword=itso4you
agentSslKeyStore=E:\\SSL\\ATLANTA\\keystore.jks
agentSslKeyStorePassword=itso4you
```

Figure 6-46 The agent.properties file for the ITSO Atlanta store File Transfer Edition agent

Figure 6-47 illustrates the agent.properties file for the ITSO Enterprise Chicago store File Transfer Edition agent that has been updated with the SSL properties.

```
1 agent.properties
#
#Mon Jun 27 07:36:26 EDT 2011
agentQMgr=STORE.CHICAGO.QM
agentQMgrPort=1414
agentDesc=Agent for ITSO Chicago Store
agentQMgrHost=store.chicago.itso.com
agentQMgrChannel=ITSO.SSL.SVRCONN
agentName=STORE.CHICAGO.QM
commandPath=E:\\Scripts\\CHICAGO
agentSslCipherSpec=RC4_MD5_US
agentSslTrustStore=E:\\SSL\\CHICAGO\\truststore.jks
agentSslTrustStorePassword=itso4you
agentSslKeyStore=E:\\SSL\\CHICAGO\\keystore.jks
agentSslKeyStorePassword=itso4you
```

Figure 6-47 The agent.properties file for the ITSO Chicago store File Transfer Edition agent

Figure 6-48 on page 312 illustrates the agent.properties file for the ITSO Enterprise NEWYORK store File Transfer Edition agent that has been updated with the SSL properties.

```

1 agent.properties
#
#Mon Jun 27 07:36:59 EDT 2011
agentQMgr=STORE.NEWYORK.QM
agentQMgrPort=1414
agentDesc=Agent for ITSO NewYork Store
agentQMgrHost=store.newyork.itso.com
agentQMgrChannel=ITSO.SSL.SVRCONN
agentName=STORE.NEWYORK.QM
commandPath=E:\\Scripts\\NEWYORK
agentSslCipherSpec=RC4_MD5_US
agentSslTrustStore=E:\\SSL\\NEWYORK\\truststore.jks
agentSslTrustStorePassword=itso4you
agentSslKeyStore=E:\\SSL\\NEWYORK\\keystore.jks
agentSslKeyStorePassword=itso4you

```

Figure 6-48 Agent.properties for NEWYORK store File Transfer Edition agent

Figure 6-49 illustrates the agent.properties file for the ITSO Enterprise warehouse File Transfer Edition agent that has been updated with the SSL properties.

```

1 agent.properties
#
#Mon Jun 27 07:36:59 EDT 2011
agentQMgr=STORE.NEWYORK.QM
agentQMgrPort=1414
agentDesc=Agent for ITSO NewYork Store
agentQMgrHost=store.newyork.itso.com
agentQMgrChannel=ITSO.SSL.SVRCONN
agentName=STORE.NEWYORK.QM
commandPath=E:\\Scripts\\NEWYORK
agentSslCipherSpec=RC4_MD5_US
agentSslTrustStore=E:\\SSL\\NEWYORK\\truststore.jks
agentSslTrustStorePassword=itso4you
agentSslKeyStore=E:\\SSL\\NEWYORK\\keystore.jks
agentSslKeyStorePassword=itso4you

```

Figure 6-49 The agent.properties file for the ITSO Enterprise warehouse agent

command.properties file

File Transfer Edition agents, commands, and administration utilities connect to the command queue manager. To configure secured SSL communication between these objects and the command queue manager, the following properties must be configured in the command.properties file:

- ▶ connectionSslCipherSpec
- ▶ connectionSslTrustStore
- ▶ connectionSslTrustStorePassword
- ▶ connectionSslKeyStore
- ▶ connectionSslKeyStorePassword
- ▶ connectionSslKeyStorePassword

Figure 6-50 illustrates the `command.properties` file for a secure connection to the queue manager `ITSO.FILE.COMMAND.QM`. Configure these properties on all servers where File Transfer Edition commands need to be executed. Also, copy the truststore and keystore to these servers that were created for MQ Explorer in the server `fileadmin.itso.com` (Example 6-80 on page 302).

```
1 command.properties
#Fri Jun 10 05:51:56 EDT 2011
connectionQMGrChannel=ITSO.SSL.SVRCONN
connectionQMGrPort=1415
connectionQMGrHost=fileadmin.itso.com
connectionQMGr=ITSO.FILE.COMMAND.QM
connectionSslCipherSpec=RC4_MD5_US
connectionSslTrustStore=E:\\SSL\\EXPLORER\\truststore.jks
connectionSslTrustStorePassword=itso4you
connectionSslKeyStore=E:\\SSL\\EXPLORER\\keystore.jks
connectionSslKeyStorePassword=itso4you
```

Figure 6-50 `command.properties`

coordination.properties file

File Transfer Edition agents, commands, and administration utilities connect to the coordination queue manager to configure the secured SSL communication `coordinationSslCipherSpec`, `coordinationSslTrustStore`, `coordinationSslTrustStorePassword`, `coordinationSslKeyStore`, and `coordinationSslKeyStorePassword`. The properties must be configured in the `coordination.properties` file.

Figure 6-51 illustrates the `command.properties` file for a secure connection to the queue manager `ITSO.FILE.COORDINATION.QM`. Configure these properties on all servers where the File Transfer Edition commands need to be executed. Also, copy the truststore and keystore to these servers that were created for MQ Explorer in the server `fileadmin.itso.com` (Example 6-80 on page 302).

```
1 coordination.properties
#Fri Jun 10 05:51:23 EDT 2011
coordinationQMGr=ITSO.FILE.COORDINATION.QM
coordinationQMGrHost=fileadmin.itso.com
coordinationQMGrChannel=ITSO.SSL.SVRCONN
coordinationQMGrPort=1416
coordinationSslCipherSpec=RC4_MD5_US
coordinationSslTrustStore=E:\\SSL\\EXPLORER\\truststore.jks
coordinationSslTrustStorePassword=itso4you
coordinationSslKeyStore=E:\\SSL\\EXPLORER\\keystore.jks
coordinationSslKeyStorePassword=itso4you
```

Figure 6-51 `coordination.properties`

After the configuration is complete, restart all the queue managers, start the File Transfer Edition agents, and verify the agent logs for successful start-up.

6.9.3 Testing the SSL configuration

To test the SSL configuration, we test the file transfers. Create a file transfer request by placing the Inventory and trigger files in the File Transfer Edition agent monitoring location:

1. Place the `ITSOATL.INVENTORY.csv` file and the `ITSOATL.INVENTORY.go` file in the directory `E:\ITSO.INVENTORY.FILES\ATLANTA` of the server `store.atlanta.itso.com`.
2. Place the `ITSOCHI.INVENTORY.csv` and `ITSOCHI.INVENTORY.go` files in the directory `E:\ITSO.INVENTORY.FILES\CHICAGO` of the server `store.chicago.itso.com`.
3. Place the `ITSONEW.INVENTORY.csv` and `ITSONEW.INVENTORY.go` files in the directory `E:\ITSO.INVENTORY.FILES\NEWYORK` of the server `store.newyork.itso.com`.

After the presence of the trigger file is determined by the File Transfer Edition monitors, the monitoring tasks schedule an inventory and trigger file transfer from each store to the warehouse. Figure 6-52 illustrates the file transfers that have been completed over the SSL configuration.

Transfer Log				
	Source	Destination	Completion State	Started (America/New_York)
	STORE.CHICAGO.QM		Successful	7/11/11 12:46:05 AM EDT
	STORE.CHICAGO.QM	WAREHOUSE.AGENT.QM	Successful	7/11/11 12:46:08 AM EDT
	STORE.CHICAGO.QM	WAREHOUSE.AGENT.QM	Successful	7/11/11 12:46:09 AM EDT
	STORE.ATLANTA.QM		Successful	7/11/11 12:46:28 AM EDT
	STORE.ATLANTA.QM	WAREHOUSE.AGENT.QM	Successful	7/11/11 12:46:30 AM EDT
	STORE.ATLANTA.QM	WAREHOUSE.AGENT.QM	Successful	7/11/11 12:46:31 AM EDT
	STORE.NEWYORK.QM		Successful	7/11/11 12:46:38 AM EDT
	STORE.NEWYORK.QM	WAREHOUSE.AGENT.QM	Successful	7/11/11 12:46:40 AM EDT
	STORE.NEWYORK.QM	WAREHOUSE.AGENT.QM	Successful	7/11/11 12:46:40 AM EDT

Figure 6-52 File transfer results with SSL configuration

A simple way to check whether SSL is configured is to deliberately mismatch your WebSphere MQ File Transfer Edition and queue manager channel cipher suites or empty the location of the truststore in the properties file.

We remove the `coordinationSslTrustStore` property from the `coordination.properties` file. This action causes the `ftelistagents` command to fail with `MQRC_JSSE_ERROR`. Example 6-110 illustrates the failure when the valid certificates are not available to the user for issuing administration commands.

Example 6-110 The `ftelistagents` command fails with MQ error code 2397

```
C:\>ftelistagents
5655-U80, 5724-R10 Copyright IBM Corp. 2008, 2011. ALL RIGHTS RESERVED
BFGCL0002E: A messaging problem prevented the command from completing successfully. The WebSphere MQ completion code was 2, and the reason code was 2397. A connection could not be established to queue manager ITSO.FILE.COORDINATION.QM, on host 'ftesadmin.itso.com' using port 1416 and channel ITSO.SSL.SVRCONN.
```

With the implementation of secure file transfer with WebSphere MQ File Transfer Edition, ITSO Enterprise addressed both of its requirements.

6.10 Summary

In this chapter, we discussed the Managed File Transfer and how it can overcome the limitations of existing FTP applications. We introduced the IBM Managed File Transfer product, WebSphere MQ File Transfer Edition.

We later discussed the problems of ITSO Enterprise with FTP. ITSO Enterprise evaluated the Enterprise Messaging and File Backbone pattern using File Transfer Edition. This Managed File Transfer implementation provided a solution for the following business requirements:

- ▶ Checkpoint and restart of file transfer
- ▶ End-to-end automation
- ▶ Audit file transfer
- ▶ Centralized administration
- ▶ Secure file transfer

We described the technical implementation of base infrastructure using WebSphere MQ and WebSphere MQ File Transfer Edition. We described the implementation and testing steps for the core business requirements.

With the fulfillment of the core business requirement for enterprise file transfer, ITSO Enterprise can look forward to rapid SOA integration opportunities with its existing Managed File Transfer systems.

Enterprise file transfer has come a long way from the FTP server; today's Managed File Transfer solutions offer better manageability, true workflow, powerful automation, and more. Managed File Transfer is the future when compared to the return on investment of homegrown solutions.

Archived

Evolve to SOA pattern

This chapter describes the Evolve to Service-Oriented Architecture (SOA) pattern using the Smart SOA approach.

In this chapter, we discuss the supply aspect of ITSO Enterprise, which is facing challenges in the business process and connectivity with external partners areas.

In this chapter, we introduce IBM WebSphere Message Broker and demonstrate how it can be used to provide service virtualization by providing additional levels of integration through an enterprise service bus (ESB).

We cover the following topics in this chapter:

- ▶ 7.1, “Introduction to the business scenario” on page 318
- ▶ 7.2, “SOA solution” on page 323
- ▶ 7.3, “Benefits” on page 325
- ▶ 7.4, “Technical implementation ” on page 326
- ▶ 7.5, “Summary” on page 351

How to recreate these scenarios: You can download the configuration files and programs that are used in this chapter from the ITSO Redbooks FTP site. You can use these files to recreate these scenarios in your environment. For download instructions, refer to Appendix A, “Additional material” on page 401.

7.1 Introduction to the business scenario

In Chapter 6, “Enterprise Messaging and File Backbone pattern” on page 233, we saw how MQ File Transfer Edition helped our fictional enterprise ITSO Enterprise to exchange information between the stores and the warehouse in a reliable and secure manner. In this chapter, we take a closer look at the ITSO Enterprise warehouse that serves as the centralized supply hub for all the stores for their inventory needs.

We first explore the existing IT solution at the warehouse and understand the limitations of the current scenario. Later, we present how IBM SOA connectivity products are used not only to resolve the current problems but also to provide newer capabilities and introduce agility to the enterprise.

ITSO Enterprise has a centralized warehouse, which manages the inventory for all the stores. It uses the off-the-shelf Inventory Management Software InvITSO and a few custom ITSO Enterprise-written programs. Having centralized software helps ITSO Enterprise to achieve economics of scale and lower IT costs.

Implementing InvITSO was a significant cost, but it has been worth all the effort and money. ITSO Enterprise has customized InvITSO extensively over the years and it now contains significant business knowledge that is built into it. It is too difficult to replace the InvITSO Software. ITSO Enterprise uses InvITSO for a variety of purposes:

- ▶ Maintaining an optimal widget inventory at the stores and the warehouse
- ▶ Creating orders for the widget suppliers
- ▶ Receiving widgets into the warehouse or other stores.
- ▶ Picking, packing, and shipping widgets from the warehouse
- ▶ Keeping track of widget sales and inventory levels across the stores

ITSO Enterprise uses custom programs to place the orders with the widget suppliers. ITSO Enterprise has negotiated long-term supply contracts with various suppliers for widgets, which helps ITSO Enterprise achieve shorter lead times and lower supply costs. ITSO Enterprise does not have any mechanism to keep track of the orders online. To find the status of the orders, ITSO Enterprise has to call the supplier and get the information.

Figure 7-1 on page 319 shows the current business setting.

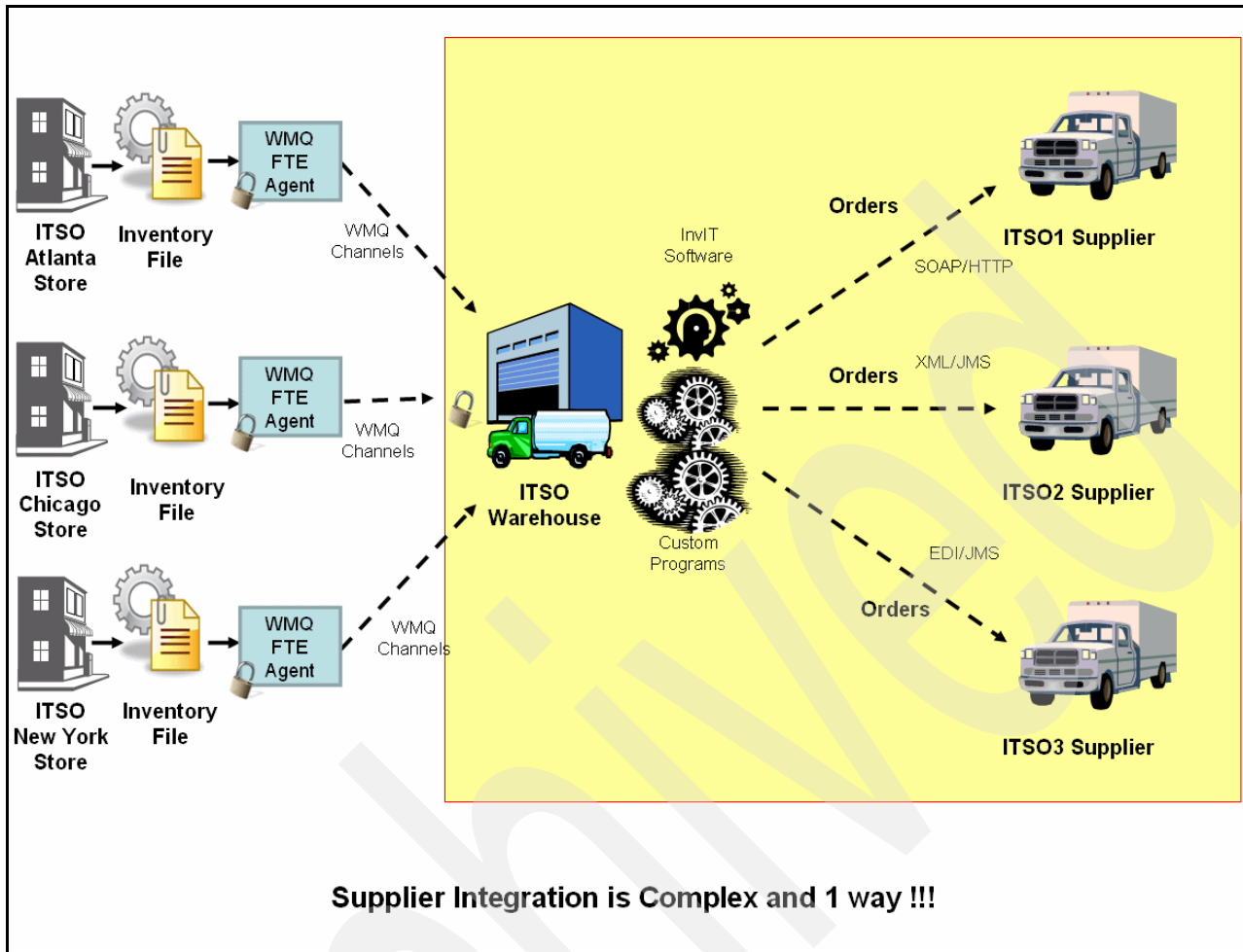


Figure 7-1 Overview of the current supplier integration

7.1.1 Current implementation

The stores used to send their inventory data file using FTP to the warehouse. ITSO Enterprise recently introduced MQ File Transfer Edition for transferring the inventory data from the stores to the warehouse in a secured and reliable manner.

The stores now send the inventory data to the warehouse daily via the File Transfer Edition. During the night, the inventory data for each store is collated and fed into the Inventory Management Software called InvITSO. InvITSO uses the daily widget inventory data that is sent by all the stores as input to make the decisions. It uses the various business rules and the inventory data to determine the supplies required by the warehouse. It also creates the supplier-specific work order output files. These files are then used by ITSO Enterprise-built custom programs to create the orders with the various suppliers.

Details of the current implementation

The stores send their daily inventory report to the warehouse via File Transfer Edition each night. The file is named in the format `store_id_mmdyy.txt`. It is a txt file in a comma-separated value (csv) format:

Store Location Id, Widget Id, Current Inventory

The various stores have location IDs that have been shared with the suppliers. This ID is used to uniquely identify the store locations by both the ITSO Enterprise warehouse and the ITSO Enterprise suppliers.

The warehouse receives files throughout the day from the stores. The File Transfer Edition solution also combines all the store files into one big file. This file is then passed to the InvITSO Software in the warehouse. To create the orders for the suppliers, InvITSO performs these steps:

1. InvITSO takes the csv input file that contains the following columns:
Store Location Id, Widget Id, and Current Inventory
2. It calculates the current widget inventory level at each store.
3. It determines if a reorder is required for the widget and the optimal reorder quantity per the applicable business rules. It also determines the best supplier for a given widget and store combination. The output of the program is a supplier-specific csv file that contain Store Location Id, Widget Id, and Quantity.

The name of the file is *supplier_id.txt*. The following sample shows the file content:

```
34,2001,3000
34,2005,60
11,184,345
```

The warehouse has multiple custom order programs that take the supplier-specific csv output file from the InvITSO Inventory Software and then uses it to create the order for the suppliers. Because the suppliers use differing message formats and transmission protocols, separate programs need to be run to place the orders. ITSO Enterprise has multiple suppliers, but we look at three representative suppliers.

Table 7-1 ITSO suppliers

Supplier	Message format	Transport protocol
ITSO1	SOAP	HTTP
ITSO2	XML	Java Message Service (JMS)
ITSO3	EDI	JMS

Let us now discuss the suppliers individually.

For ITSO1, the orders are placed in SOAP format over HTTP. Example 7-1 shows the sample format.

Example 7-1 The orders are placed in SOAP format over HTTP

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:q0="http://www.itso.com/warehouse/supplier/OrderStatusService"
xmlns:xalan="http://xml.apache.org/xslt">
  <soapenv:Body>
    <q0:submitPORequest>
      <clientID>456809</clientID>
      <orderID>34</orderID>
      <orderDate>20110705</orderDate>
      <orderDetails>
        <partNumber>705</partNumber>
        <location>33</location>
        <quantity>800</quantity>
      </orderDetails>
    </q0:submitPORequest>
  </soapenv:Body>
</soapenv:Envelope>
```

```

</orderDetails>
<orderDetails>
  <partNumber>209</partNumber>
  <location>30</location>
  <quantity>450</quantity>
</orderDetails>
<orderDetails>
  <partNumber>2356</partNumber>
  <location>11</location>
  <quantity>100</quantity>
</orderDetails>
</q0:submitPORequest>
</soapenv:Body>
</soapenv:Envelope>

```

For ITSO2, the orders are placed as XML via MQ queue. Example 7-2 shows the sample format.

Example 7-2 The orders are placed as XML via MQ queue

```

<?xml version="1.0" encoding="UTF-8"?>
<bo:SubmitOrder xmlns:bo="http://www.itso2.com/Orders/input/data"
xmlns:xalan="http://xml.apache.org/xslt">
  <submitOrderStatusRequest>
    <clientID>456809</clientID>
    <orderID>35</orderID>
    <orderDate>20110705</orderDate>
    <orderDetails>
      <partNumber>1974</partNumber>
      <locationID>43</locationID>
      <quantity>2600</quantity>
    </orderDetails>
    <orderDetails>
      <partNumber>87</partNumber>
      <locationID>47</locationID>
      <quantity>213</quantity>
    </orderDetails>
    <orderDetails>
      <partNumber>214</partNumber>
      <locationID>11</locationID>
      <quantity>566</quantity>
    </orderDetails>
  </submitOrderStatusRequest>
</bo:SubmitOrder>

```

For ITSO3, the orders are expected in electronic data interchange (EDI) format over the MQ queue. Example 7-3 shows the sample format.

Example 7-3 The orders are expected in the EDI format over the MQ queue

```

P0~33###CL~456809##0D~20110705***
L~01###34###2001###3000###***
L~02###34###2005###60###***
L~03###11###184###345###***
P0***

```

Products used

We describe the products that are used in the stores and the warehouse.

Stores

The daily inventory report is generated from the stores' point of sale (POS) devices. This file is sent to the warehouse using MQ File Transfer Edition.

Warehouse

Multiple applications are in use at the warehouse:

- ▶ The Inventory Software manages all the inventory-related calculations.
- ▶ The custom Java programs create the orders.
- ▶ Before the upgrade to MQ File Transfer Edition, the warehouse also used the FTP server to receive all the files from the stores and a UNIX shell program to combine all the store FTP files.

7.1.2 Technical challenges

The ITSO Enterprise IT department identified the following challenges in the current scenario:

- ▶ The suppliers require separate message formats (EDI, XML, and SOAP) over separate protocols, such as FTP, JMS, and SOAP. The current program has to work with both the transformation and protocol variations. Due to these variations, the program is difficult to maintain, because it has to be consistently updated to meet the supplier requirements. ITSO Enterprise needs to handle the variations in a better manner.
- ▶ Order tracking with suppliers is manual and takes up a lot of the ITSO Enterprise staff's time. An online automated process will be faster and cost-effective.
- ▶ ITSO plans to sell new varieties of widgets and will have to integrate with the new suppliers. Adding new suppliers forces ITSO Enterprise to change the Order programs. Any change at the suppliers' end forces change in the Order program. Keeping the Order application up-to-date puts a great strain on the ITSO Enterprise IT resources, because even a single day of outage means unsatisfied widget customers.
- ▶ Centralized management of the supplier orders is not available. Currently, to provide a summarized view to higher management, ITSO Enterprise needs to manually collate the data across the suppliers.
- ▶ Maintaining consistency across the various Order programs is difficult. Recently, the ITSO Enterprise business department wanted to implement an auditing mechanism for the orders, but the IT team discovered that it was difficult to make similar changes in all the Order programs.

7.1.3 Evolving to service-oriented architecture (SOA)

Chapter 2, "Service Enablement pattern" on page 23 introduces the principles of the ESB by exposing services that can be transparently consumed by any of the channels, such as Customer Service Representatives (CSR), web, and so on.

ITSO needs a more sophisticated, manageable infrastructure with these capabilities:

- ▶ Support for high volumes of individual interactions
- ▶ Support for more established mediation patterns, such as routing, protocol conversion, data transformation, logging, and so on
- ▶ Integration with existing infrastructures

- ▶ Support for enterprise-level qualities of service (QoS)

In this chapter, we describe how the ESB provides the unifying concept for this infrastructure.

The current warehouse IT support consists of multiple stand-alone programs: the InvITSO Software and the recently acquired File Transfer Edition solution. To overcome the problems associated with the current solution, ITSO Enterprise will introduce the ESB, which can use the existing MQ infrastructure. The IBM middleware product WebSphere Message Broker V7 was proposed to realize the ESB.

WebSphere Message Broker is a platform-independent-based ESB that provides universal connectivity. It can be used to integrate disparate applications and is designed to transform various formats of data between any types of applications using a number of supported communications protocols or distribution methods. It is used where there is a need for high performance and complex integration patterns.

WebSphere Message Broker V7.0 offers simplicity and productivity in terms of developing and managing the WebSphere Message Broker environment. WebSphere Message Broker plays a critical role in SOA and offers a wide range of SOA scenarios in which it can be integrated. The dynamic operational management of WebSphere Message Broker enables administrators to effectively understand and modify broker behavior, which thus enables them to respond quickly to business requirements. WebSphere Message Broker is supported on a large range of platforms and environments.

The processing logic in WebSphere Message Broker is implemented using message flows. Through message flows, messages from business applications can be transformed and routed to other business applications. Message flows are created by connecting nodes together. A wide selection of built-in nodes are provided with WebSphere Message Broker. These nodes perform tasks that are associated with message routing, transformation, and enrichment. The base capabilities of WebSphere Message Broker are enhanced by SupportPacs that provide a wide range of additional enhancements.

WebSphere Message Broker V7 provides patterns that offer these benefits:

- ▶ Gives guidance in implementing solutions
- ▶ Increases development efficiency because resources are generated from a set of predefined templates
- ▶ Improves quality through asset reuse and the common implementation of functions, such as error handling and logging

The ESB infrastructure also ensures that ITSO Enterprise IT is better aligned for future changes, such as adding new suppliers or adapting to changes in suppliers easily.

7.2 SOA solution

The complicated and difficult to maintain separate Order programs can be replaced with WebSphere Message Broker flows. We will exploit the strengths of WebSphere Message Broker to easily take care of the message formats (through transformations) and technology variations (through protocol switching). We will also use the built-in pattern to get a head start with the development.

A WebSphere Message Broker message flow picks up the Inventory Software InvITSO responses from a directory. It creates the individual supplier order messages and uses the supplier's preferred protocol to send the message. The message flow program is generic and

can be used for all the suppliers. The variation in the messages and protocols is handled using the externalized configuration that is kept in the database in Extensible Stylesheet Language Transformation (XSLT) format. In summary, the WebSphere Message Broker flow does the following tasks:

1. It parses the InvITSO output .csv files using the message set and converts it into XML messages. This .csv to XML conversion enables the easier transformation of the order data in multiple formats using XSLT.
2. The supplier name can be fetched from the file name. It then accesses the database table SUPPLIER to get the flow details. The SUPPLIER table has the columns SUPPLIERID, SUPPLIERNAME, FORWARDXSL, and ENDPOINT.
3. The stylesheet name FORWARDXSL is used to convert the csv information to the supplier order.
4. The ENDPOINT information is used to handle the technology protocol.
5. It logs the Order request in the order-related tables: ORDER_HEADER and ORDER_DETAILS.
6. It accesses the order service.
7. It logs the service response in the ORDER_STATUS table: ORDERID, STATUS, and TIMESTAMP.

This approach also ensures that the stores and suppliers are not affected by ITSO Enterprise IT changes. This process also uses the existing asset, the InvITSO Inventory Software.

ITSO IT can implement these changes with minimum impact to its business. See Figure 7-2 on page 325 for a description of this solution.

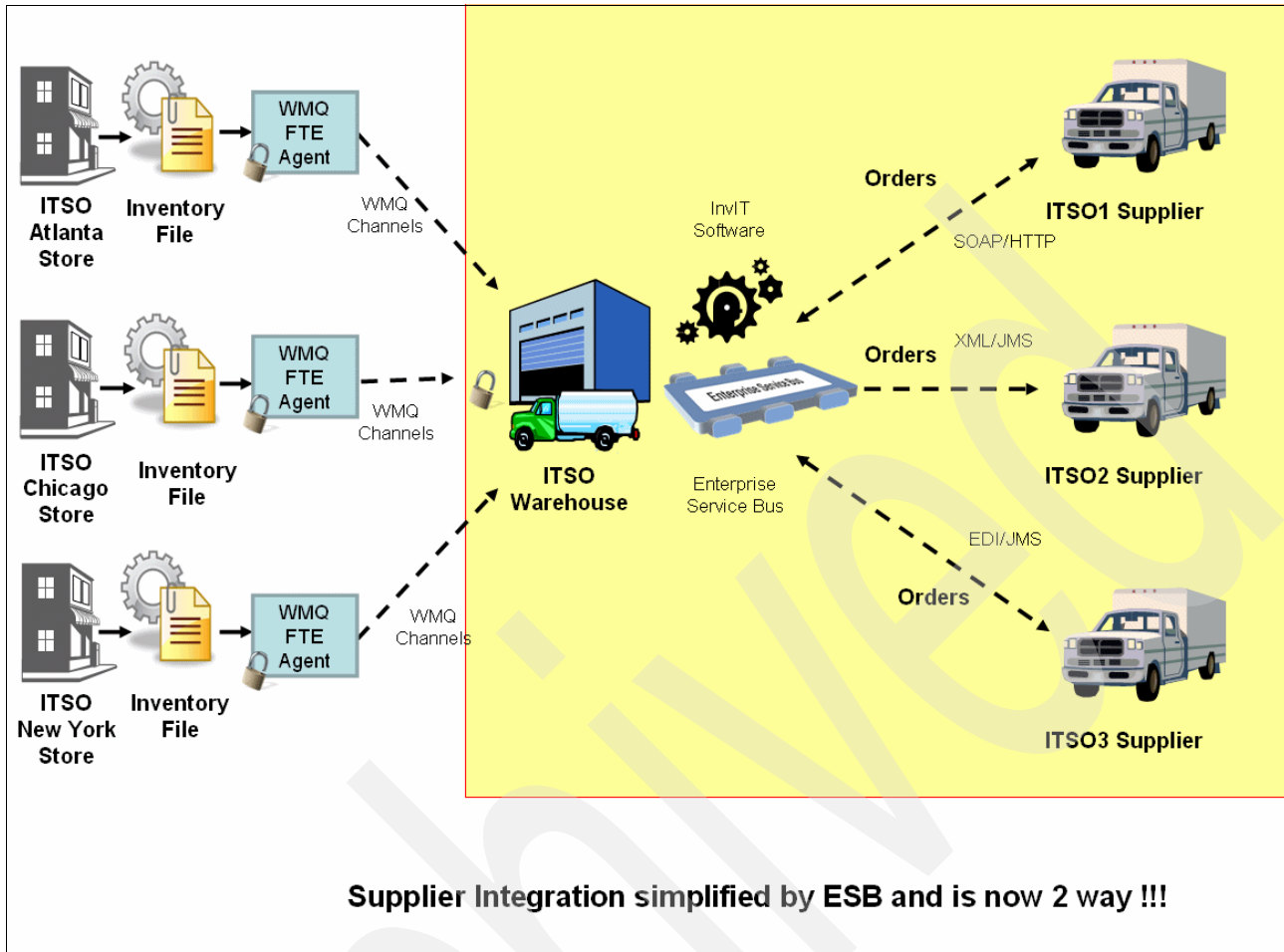


Figure 7-2 Supplier integration for ITSO Enterprise

To overcome the problem of the determination of the order status, WebSphere Message Broker flow can expose a Web service which can be used by the suppliers to report the order status. This order status is stored in the database and can be used to track the supplier orders. The flow is designed to expose a Web service to accept orderid and status and to insert this status into the ORDER_STATUS table: ORDERID, STATUS, and TIMESTAMP.

7.2.1 Products utilized to implement this pattern

The ESB solution was implemented using IBM WebSphere Message Broker Version 7.0.0.2.

To get more information about this product, see this website:

<http://www.ibm.com/software/integration/wbmessagebroker/>

7.3 Benefits

Using the Smart SOA solution that is implemented by ITSO Enterprise offers these benefits:

- ▶ Adding new suppliers is easy, because all the configuration is now externalized from the code to the database. This design enables easy configuration without touching any code or needing any redeployment.

- ▶ Centralized management of the ordering is possible. Due to the error-handling capabilities, retry strategy, security, and auditing can be implemented at the enterprise level.
- ▶ Order tracking was exposed as a service, which helped ITSO Enterprise to track orders more easily.
- ▶ Existing assets are reused. The huge investment that was made in InvITSO can be reused even after all the changes have been made to the warehouse applications.
- ▶ This design is nondisruptive to the suppliers. The suppliers are unaffected by changes at the warehouse.
- ▶ WebSphere Message Broker provides the features of enterprise-quality software, such as high availability, security, and reliability.

7.4 Technical implementation

The WebSphere Message Broker message flow is used to implement the Order creation process. In this section, we describe this implementation step-by-step.

How to recreate these scenarios: You can download the configuration files and programs that are used in this chapter from the ITSO Redbooks FTP site. You can use these files to recreate these scenarios in your environment. For download instructions, refer to Appendix A, “Additional material” on page 401.

7.4.1 Database setup

This sample assumes that the warehouse database name is ITSO and the various tables were created using the script that is shown in Example 7-4.

Example 7-4 Database setup script

```

drop table SUPPLIER
/
create table SUPPLIER
(SUPPLIERID INT NOT NULL,
SUPPLIERNAME VARCHAR(128),
FORWARDXSL VARCHAR(255),
ENDPOINT VARCHAR(255))
/
drop table ORDER_HEADER
/
create table ORDER_HEADER
(ORDERID INT NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1,
NO CACHE),
SUPPLIERID VARCHAR(10),
ORDERDATE DATE NOT NULL DEFAULT CURRENT_DATE)
/
drop table ORDER_DETAILS
/
create table ORDER_DETAILS
(ORDERID INT NOT NULL,
WIDGETID VARCHAR(30),
QUANTITY VARCHAR(10),
STOREID VARCHAR(10))

```

```

/
drop table ORDER_STATUS
/
create table ORDER_STATUS
(ORDERID INT NOT NULL,
STATUS VARCHAR(255),
UPDATETIME TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP)
/

```

This sample assumes that the queue manager name for the warehouse is QMWAREHOUSE. Run the script `inserts.sql` (Example 7-5) to insert the metadata for the supplier.

Example 7-5 inserts.sql

```

delete from SUPPLIER
/
insert into SUPPLIER (supplierid, suppliername, forwardxsl, endpoint) values
(2344,'ITS01','ITS01_Order.xsl','http://www.itso3.com:7080/Orders/web')
/
insert into SUPPLIER (supplierid, suppliername, forwardxsl, endpoint) values
(90045,'ITS02','ITS02_Order.xsl','queue://QMWAREHOUSE/ITS0.JMS.90045.QUEUE')
/
insert into SUPPLIER (supplierid, suppliername, forwardxsl, endpoint) values
(90019,'ITS03','ITS03_Order.xsl','queue://QMWAREHOUSE/ITS0.EDI.90019.QUEUE')
/

```

7.4.2 MQ setup

Run the WebSphere MQ queue creation commands that are shown in Example 7-6 against the warehouse queue manager.

Example 7-6 MQ setup

```

DEFINE QLOCAL('ITSO.WAREHOUSE.ORDERS.SOAP.RESPQ') MAXDEPTH(5000) REPLACE
DEFINE QLOCAL('ITSO.WAREHOUSE.ORDERS.ERRORQ') MAXDEPTH(5000) REPLACE
DEFINE QLOCAL('ITSO.JMS.90045.QUEUE') MAXDEPTH(5000) REPLACE
DEFINE QLOCAL('ITSO.EDI.90019.QUEUE') MAXDEPTH(5000) REPLACE

```

7.4.3 Broker setup

Perform the following steps for the broker setup:

1. Create an ODBC entry for the database (based on your operating system) and then run the following command against the broker:

```

mqsisetdbparms <broker name> -n <odbc data source> -u <db2 user id> -p <db2
password>

```

For example:

```

mqsisetdbparms warehouse -n ITSO -u db2admin -p passw0rd

```

The message flow assumes that the InvITSO Software output is in the directory `C:\1z01\orders\suppliers\`.

2. Create the Database Definition ITSO in the Message Broker toolkit pointing to the ITSO database. Ensure that the connection works using the Test Connection in the wizard.

7.4.4 Message flow development with the Message Broker Toolkit

We create a new message set. This message set will be used in the message flow project to convert the orders from .csv to XML format.

Creating a new message set project

Perform the following steps to create a new message set project:

1. For the Message set name, type CSV2XML, and for the Message set project name, type ITSO_Warehouse_CreateOrder_MessageSet. Click **Next**. See Figure 7-3.

The screenshot shows a 'New Message Set' dialog box with the following configuration:

- Message set name:** CSV2XML
- Message set project name:** ITSO_Warehouse_CreateOrder_MessageSet
- Project location:** Use default. Directory: C:\redbook\MbWorkspace1\ITSO_Warehouse_CreateOrder_Mes: (with a 'Browse...' button).
- Copy message set contents from another message set:** Message set: <create message set with no message defintions>

Navigation buttons at the bottom: < Back, Next >, Finish, Cancel.

Figure 7-3 Creating a new message set for csv to XML transformation

2. On the New Message Set window, choose **XML documents (general)** in the type of message data drop-down list box. Select **Text data (for example, CSV, SWIFT or HL7)** and **CSV - Comma Separated Values** in the Text messaging standard and click **Finish**. See Figure 7-4.

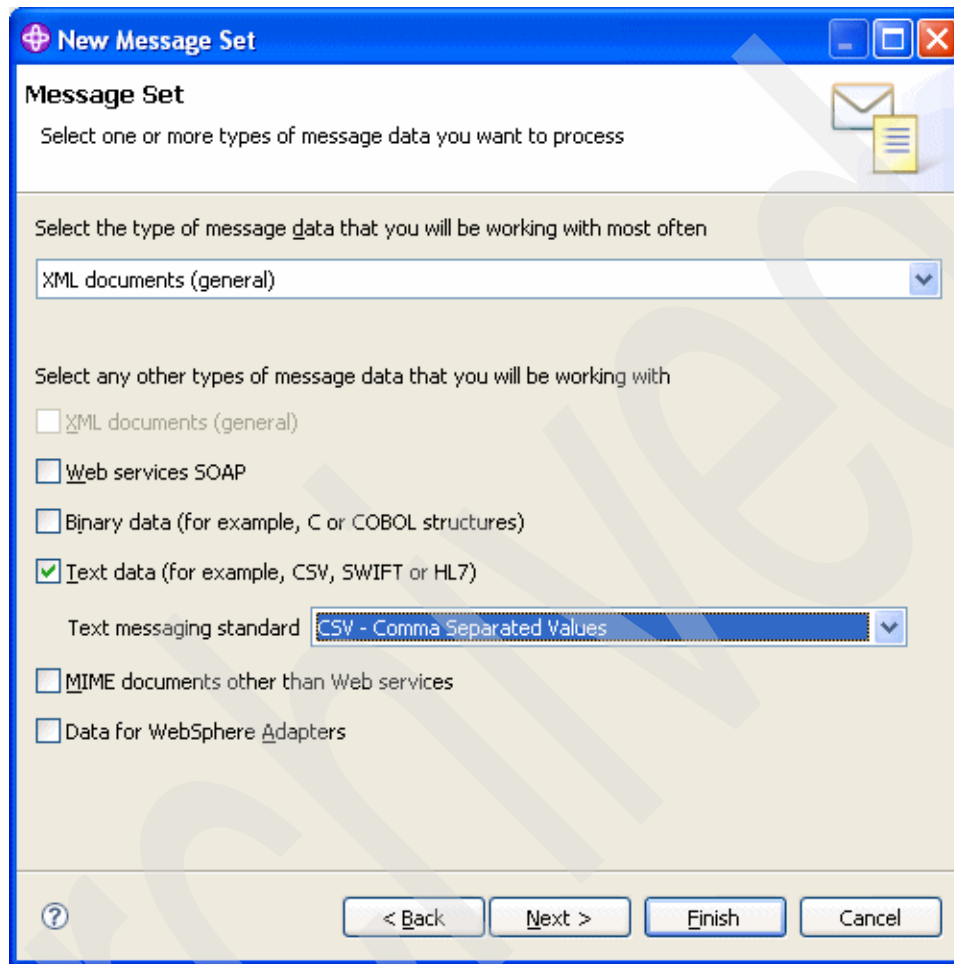


Figure 7-4 Selecting the type of message data

- The generated message scheme definition file `csv.mxsd` is edited to reflect the file that is generated by the InvITSO Software by adding the `storeId`, `widgetId`, and `quantity` columns. Go to **Message Definitions** → **csv.mxsd**. Right-click and select **Open** to get the Message Definition Editor. See Figure 7-5.

Structure	Type	Min Occurs	Max Occurs
[-] csv.mxsd			
[-] Messages			
[-] CSV	CSVInput		
[-] orderRequest	supplier	0	-1
[-] storeId	xsd:string	1	-1
[-] widgetId	xsd:string	1	1
[-] quantity	xsd:string	1	1
[-] Types			
[-] supplier			
[-] storeId	xsd:string	1	-1
[-] widgetId	xsd:string	1	1
[-] quantity	xsd:string	1	1
[-] CSVInput			
[-] orderRequest	supplier	0	-1
[-] storeId	xsd:string	1	-1
[-] widgetId	xsd:string	1	1
[-] quantity	xsd:string	1	1
[-] Groups			
[-] Elements and Attributes			
[-] CSV	CSVInput		
[-] orderRequest	supplier	0	-1
[-] storeId	xsd:string	1	-1
[-] widgetId	xsd:string	1	1
[-] quantity	xsd:string	1	1

Figure 7-5 Schema definitions file for csv to XML conversion

This completes the creation of the message set for the .csv to XML transformation.

Creating the new message flow project to process the InvITSO output file and create the orders

Perform the following steps to create the orders:

- Go to the Pattern Explorer and select **File processing** → **Record Distribution** → **MQ one-way**.

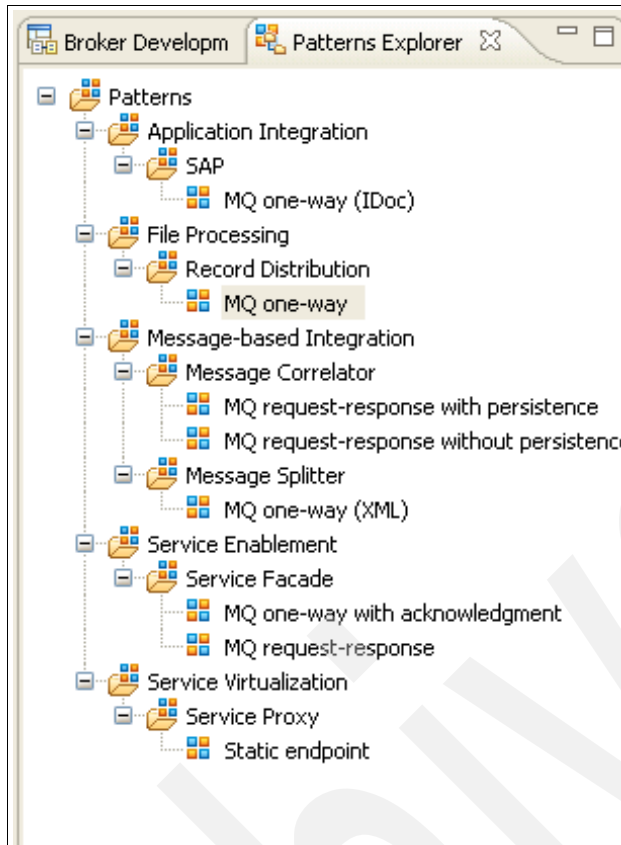


Figure 7-6 Creating new message flow

2. Click **Create New instance** to create the message flow after setting properties.
3. On the Configure Pattern Parameters window, specify the input file parameters on the Input File tab and in the Input Record Processing section. For Record detection, type Parsed (binary or text). Then, for Input message set, type CSV2XML, as shown in Figure 7-7.

Configure Pattern Parameters

Provide values for pattern parameters. Click the "Generate" button or click [here](#) to generate a pattern instance.

i Pattern workspace resources are generated successfully!

Pattern Parameters

Input file
Details of how to access the input data

Directory of input file *

File pattern *

Use FTP *

FTP configurable service *

Input record processing
Specify how file records are to be processed.

Record detection

Record length *

Delimiter type *

Record delimiter *

Input message set *

Input message type *

Input message format *

CCSID of file *

Encoding of file *

Routing

Figure 7-7 Pattern parameters

4. Click **Generate** to generate a basic flow. Then, rename the message flows to the following names:
 - ITSO_Warehouse_CreateOrder_MessageFlow.msgflow
 - ITSO_Warehouse_OrderProcessor_subflow.msgflow
 - ITSO_Warehouse_Router_subflow.msgflow
 - ITSO_Warehouse_Error_subflow.msgflow
5. Change the ITSO_Warehouse_CreateOrder_MessageFlow.msgflow to look like Figure 7-8.

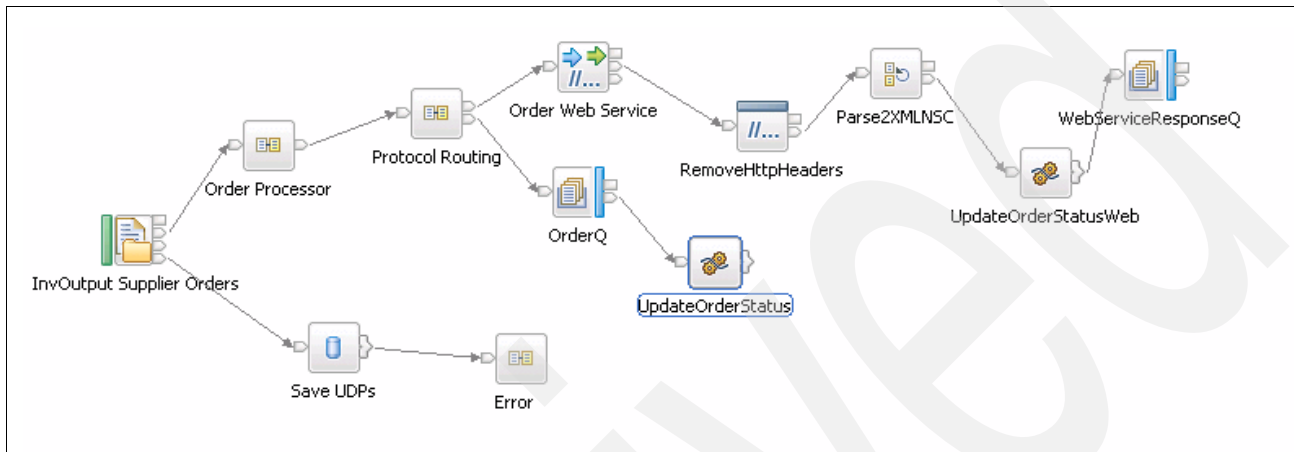


Figure 7-8 Message flow for ITSO_Warehouse_CreateOrder_MessageFlow

6. Modify the File Input Node properties, as shown in Figure 7-9. Note that on the FileInput Node Properties - InvOutput Supplier Orders tab, only the property that is shown in Figure 7-9 needs to be modified. The other properties are already set.

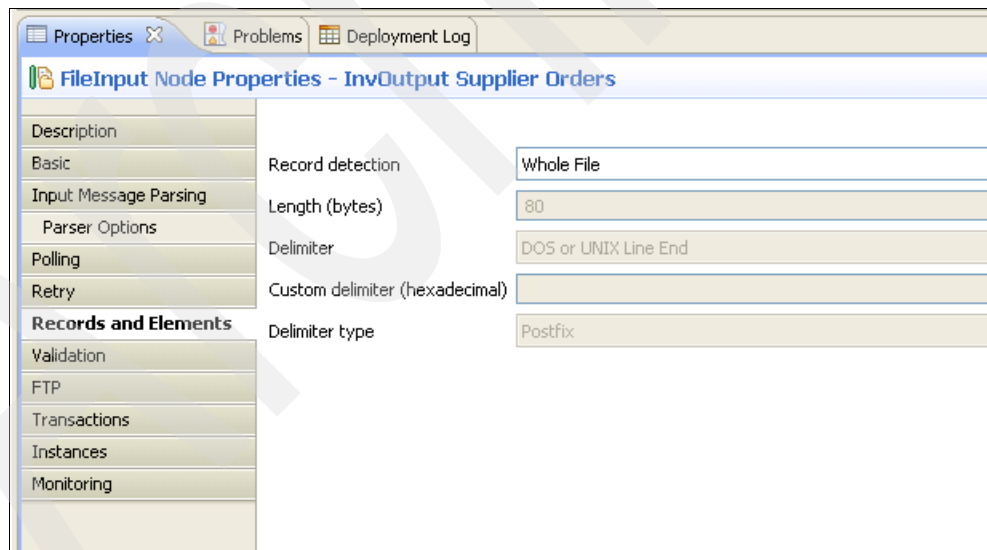


Figure 7-9 Properties for InvOutput Supplier Orders node

7. Modify the `ITSO_Warehouse_OrderProcessor_subflow`, as shown in Figure 7-10. This subflow is used to get the configuration details for the flow from the database.

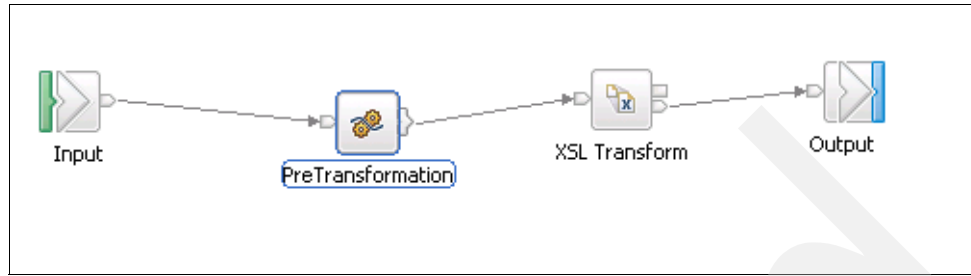


Figure 7-10 Subflow `ITSO_Warehouse_OrderProcessor_subflow`

The Extended Structured Query Language (esql) node `PreTransformation` fetches the supplier name from the file name. It then accesses the database table `SUPPLIER` to get the transformation and endpoint details. It then logs the Order request in the order-related tables: `ORDER_HEADER` and `ORDER_DETAILS`. The transformation and endpoint details are stored in the `LocalEnvironment` so that it can be used later.

The Database property must be `ITSO` and the Compute mode property of this node must be set as `LocalEnvironment` and `Message`. Example 7-7 shows the code for the `PreTransformation` esql node.

Example 7-7 Code for `PreTransformation` esql node

```

BROKER SCHEMA mqsi
CREATE COMPUTE MODULE ITSO_Warehouse_OrderProcessor_Compute
CREATE FUNCTION Main() RETURNS BOOLEAN
BEGIN
    -- CALL CopyMessageHeaders();
    DECLARE fileName CHARACTER;
    DECLARE orderInfo ROW;
    DECLARE xsl row;
    DECLARE orderId,count INTEGER;
    DECLARE curdate DATE;
    CALL CopyEntireMessage();
    DECLARE I INTEGER 1;
    DECLARE J INTEGER CARDINALITY(OutputRoot.MRM.orderRequest[]);
    SET Environment.fileName = InputLocalEnvironment.File.Name;
    SET OutputRoot.Properties.MessageFormat = 'XML';
    SET fileName = SUBSTRING(Environment.fileName BEFORE '.csv');
    INSERT INTO Database.ORDER_HEADER(SUPPLIERID,ORDERDATE)
values(fileName,CURRENT_DATE);
    SET orderInfo.Records[] = (PASSTHRU('Select p.orderid as orderid from
ORDER_HEADER as p where p.supplierid=?',fileName));
    SET count = CARDINALITY(orderInfo.Records[]);
    SET OutputRoot.MRM.orderRequest[I].clientId = '456809';
    -- Client Id is hardcoded for the ITSO;
    SET orderId = orderInfo.Records[count].*[1];
    SET Environment.orderId = orderId;
    SET OutputRoot.MRM.orderRequest[I].orderId = orderId;
    SET OutputRoot.MRM.orderRequest[I].orderDate = CAST(CURRENT_TIMESTAMP AS
CHARACTER FORMAT 'yyyyMMdd');
    WHILE I <= J DO

```

```

INSERT INTO Database.order_details
VALUES(orderId,OutputRoot.MRM.orderRequest[I].widgetid,OutputRoot.MRM.orderRequest
[I].quantity,OutputRoot.MRM.orderRequest[I].storeid);
SET I = I + 1;
END WHILE;
SET xsl.Records=(PASSTHRU('Select s.forwardxsl,s.endpoint from supplier as s
where s.supplierid=?',fileName));
SET count = CARDINALITY(xsl.Records[]);
SET OutputLocalEnvironment.XSL.StyleSheetName = xsl.Records[count].*[1];
SET OutputLocalEnvironment.Endpoint = xsl.Records[count].*[2];
RETURN TRUE;
END;
CREATE PROCEDURE CopyMessageHeaders() BEGIN
DECLARE I INTEGER 1;
DECLARE J INTEGER CARDINALITY(InputRoot.*[]);
WHILE I < J DO
SET OutputRoot.*[I] = InputRoot.*[I];
SET I = I + 1;
END WHILE;
END;
CREATE PROCEDURE CopyEntireMessage() BEGIN
SET OutputRoot = InputRoot;
END;
END MODULE;

```

- The XSL Transform node is used to transform the order file data to the message format that is required by the suppliers using XSL. Create the three XSLs, each representing a supplier.

Setup files: For the purposes of this sample, copy the XSLs from the folder transformations from the setup files that can be downloaded from the Redbooks FTP site. Refer to Appendix A, “Additional material” on page 401.

- Modify the protocol router subflow ITSO_Warehouse_Router_subflow. This subflow checks if the endpoint is JMS or a Web service and routes accordingly. See Figure 7-11.

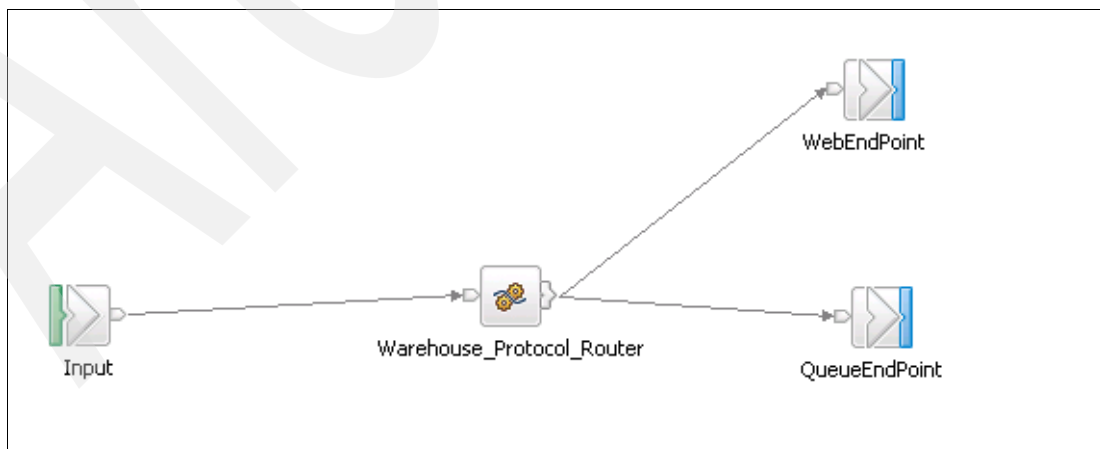


Figure 7-11 Subflow ITSO_Warehouse_Router_subflow

The esql Warehouse_Protocol_Router node is used to determine if the endpoint is the JMS queue or the web service. It then determines the endpoint details. Example 7-8 shows the esql code for the Warehouse_Protocol_Router node.

Example 7-8 esql code for the Warehouse_Protocol_Router node

```

BROKER SCHEMA mqsi
CREATE COMPUTE MODULE ITSO_Warehouse_Router_subflow
  CREATE FUNCTION Main() RETURNS BOOLEAN
  BEGIN
    DECLARE endPoint CHARACTER;
    DECLARE queueMgr1 CHARACTER;
    DECLARE queueMgr CHARACTER;
    DECLARE queueName CHARACTER;
    -- CALL CopyMessageHeaders();
    CALL CopyEntireMessage();
    SET endPoint = InputLocalEnvironment.Endpoint;
    IF CONTAINS(endPoint,'queue') THEN
      SET queueMgr1 = SUBSTRING(endPoint AFTER '//');
      SET queueMgr = SUBSTRING(queueMgr1 BEFORE '/');
      SET queueName = SUBSTRING(queueMgr1 AFTER '/');
      SET
OutputLocalEnvironment.Destination.MQ.DestinationData.queueManagerName = queueMgr;
      SET OutputLocalEnvironment.Destination.MQ.DestinationData.queueName =
queueName;
      PROPAGATE to TERMINAL 'out';
    ELSE
      SET OutputLocalEnvironment.Destination.HTTP.RequestURL = endPoint;
      PROPAGATE to TERMINAL 'out1';
    END IF;
    RETURN FALSE;
  END;
  CREATE PROCEDURE CopyMessageHeaders() BEGIN
    DECLARE I INTEGER 1;
    DECLARE J INTEGER;
    SET J = CARDINALITY(InputRoot.*[]);
    WHILE I < J DO
      SET OutputRoot.*[I] = InputRoot.*[I];
      SET I = I + 1;
    END WHILE;
  END;
  CREATE PROCEDURE CopyEntireMessage() BEGIN
    SET OutputRoot = InputRoot;
  END;
END MODULE;

```

Now that we have dealt with the subflows, we return to the main flow. The flow from here divides into the Web service and JMS endpoint. We first discuss the Web service path.

10. The HTTP Request Node is used to access the Web service of the suppliers. Configure the Order Web Service node, as shown in Figure 7-12.

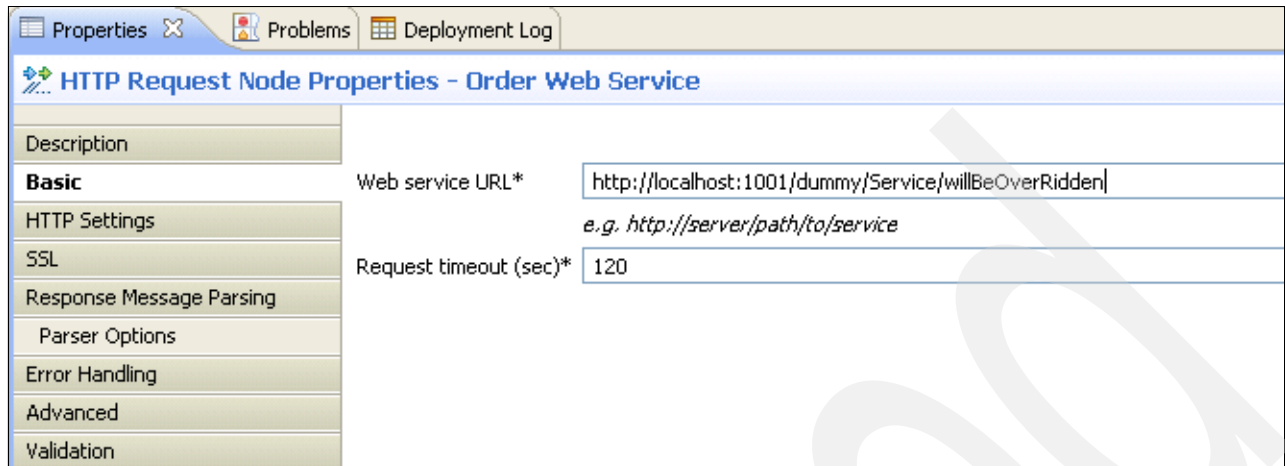


Figure 7-12 Web Service URL for the HTTP Request Node

11. Enter the Web service URL as `http://localhost:1001/dummy/Service/willBeOverRidden`. Note that this URL will be overridden by the `OutputLocalEnvironment.Destination.HTTP.RequestURL`, because we have already populated this flow variable in the previous step.
12. For the HTTP Header Node, *HTTPHeader*, configuration, choose the **Delete Header** option in HTTPInput, HTTPResponse, HTTPRequest, and HTTPReply tabs. This step is required to remove the http headers from the response of http request.
13. The Reset Content Descriptor Node, *ResetContentDescriptor*, is used to set the message back to the XMLNSC domain.
14. The ESQL node, *UpdateOrderStatusWeb*, is used to update the Order status after the supplier service is successfully called. It logs ORDERID, STATUS, and TIMESTAMP in the Order Status Table.

Example 7-9 esql code for UpdateOrderStatusWeb node

```

BROKER SCHEMA mqsi
CREATE COMPUTE MODULE ITSO_Warehouse_CreateOrder_MessageFlow_Compute
CREATE FUNCTION Main() RETURNS BOOLEAN
BEGIN
    -- CALL CopyMessageHeaders();
    CALL CopyEntireMessage();
    IF OutputRoot.XMLNSC.*:Envelope.*:Body.*:submitPOResponse.status =
'AVAILABLE' THEN
        INSERT INTO Database.ORDER_STATUS
VALUES(Environment.orderId,'Posted',CURRENT_TIMESTAMP);
    END IF;
    RETURN TRUE;
END;
CREATE PROCEDURE CopyMessageHeaders() BEGIN
    DECLARE I INTEGER 1;
    DECLARE J INTEGER;
    SET J = CARDINALITY(InputRoot.*[]);
    WHILE I < J DO
        SET OutputRoot.*[I] = InputRoot.*[I];
        SET I = I + 1;
    END WHILE;
END;

```

```

        END WHILE;
    END;
    CREATE PROCEDURE CopyEntireMessage() BEGIN
        SET OutputRoot = InputRoot;
    END;
END MODULE;

```

15. The Web service response is later logged on to the queue ITSO.WAREHOUSE.ORDERS.SOAP.RESPQ.

ITSO_Warehouse_CreateOrder_MessageFlow

Now, we discuss the JMS part of the flow:

1. The order message is placed on the OrderQ MQ Output Node. The Destination mode for OrderQ queue must be set at Destination List so that it can be assigned a value dynamically. See Figure 7-13.

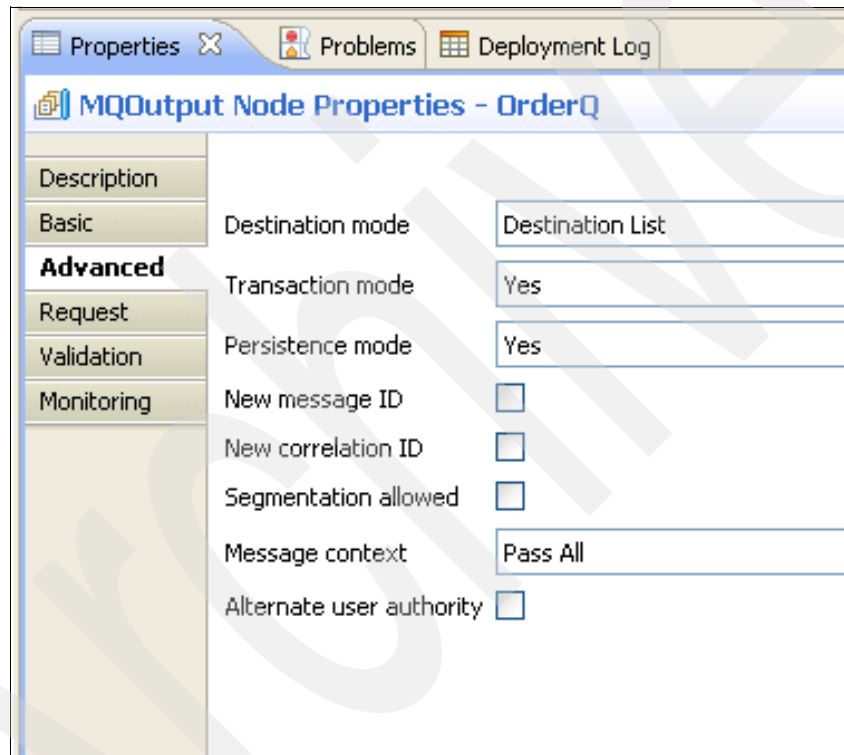


Figure 7-13 OrderQ MQ Output Node properties

2. The esql node UpdateOrderStatusJMS is used to update the order status node in the database. Example 7-10 is the code to implement this esql node.

Example 7-10 esql code for UpdateOrderStatusJMS node

```

BROKER SCHEMA mqsi
CREATE COMPUTE MODULE ITSO_Warehouse_OrderStatusUpdate
  CREATE FUNCTION Main() RETURNS BOOLEAN
  BEGIN
    -- CALL CopyMessageHeaders();
    CALL CopyEntireMessage();
    INSERT INTO Database.ORDER_STATUS
  VALUES(Environment.orderId,'Posted',CURRENT_TIMESTAMP);

```



```

    RETURN TRUE;
END;
CREATE PROCEDURE CopyMessageHeaders() BEGIN
    DECLARE I INTEGER 1;
    DECLARE J INTEGER;
    SET J = CARDINALITY(InputRoot.*[]);
    WHILE I < J DO
        SET OutputRoot.*[I] = InputRoot.*[I];
        SET I = I + 1;
    END WHILE;
END;
CREATE PROCEDURE CopyEntireMessage() BEGIN
    SET OutputRoot = InputRoot;
END;
END MODULE;

```

3. The Error subflow is generated through the pattern, which has been renamed as `ITSO_Warehouse_Error_Subflow`. It will parse `Exceptionlist` to get `Exception` data and will be added to `Message`. This message will be routed to the queue that is specified in the Error handling tab of the Pattern Parameters page or it can be modified in the `MQOutput` node of the Error subflow.

We complete the development of the `ITSO_Warehouse_CreateOrder_MessageFlow`.

In the next section, we will develop the `Web` service flow, which is used to log the `Order` status by the suppliers.

OrderStatus message flow

Perform the following steps to create the `Web` service flow (`OrderStatus`) that is used in this scenario:

1. In the `New Message Set` window (Figure 7-14 on page 340), for the `Message set name`, type `ITSOWarehouseMessageSet` and for the `Message set project name`, type `ITSO_Warehouse_OrderStatus_MessageSet`.

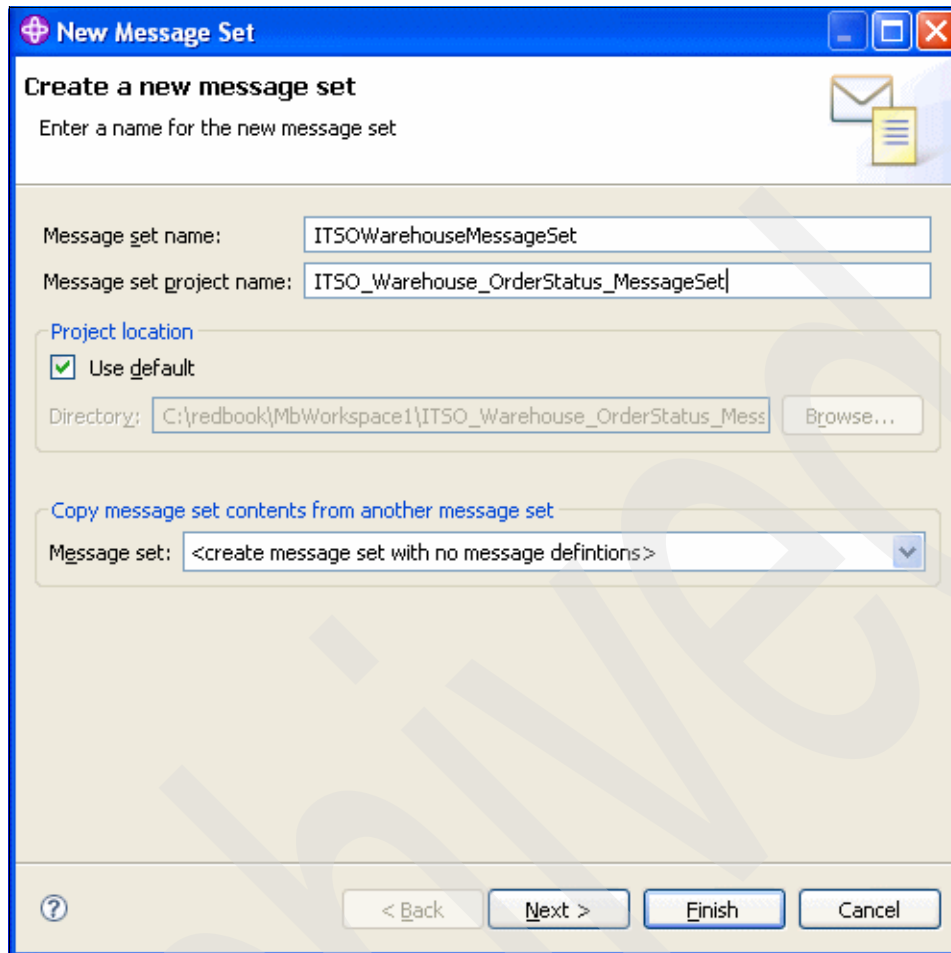


Figure 7-14 Create a new message set for the Order Status flow

2. Choose the **XML documents (general)** and **Web Services SOAP** options in the Message Set window, as shown in Figure 7-15.

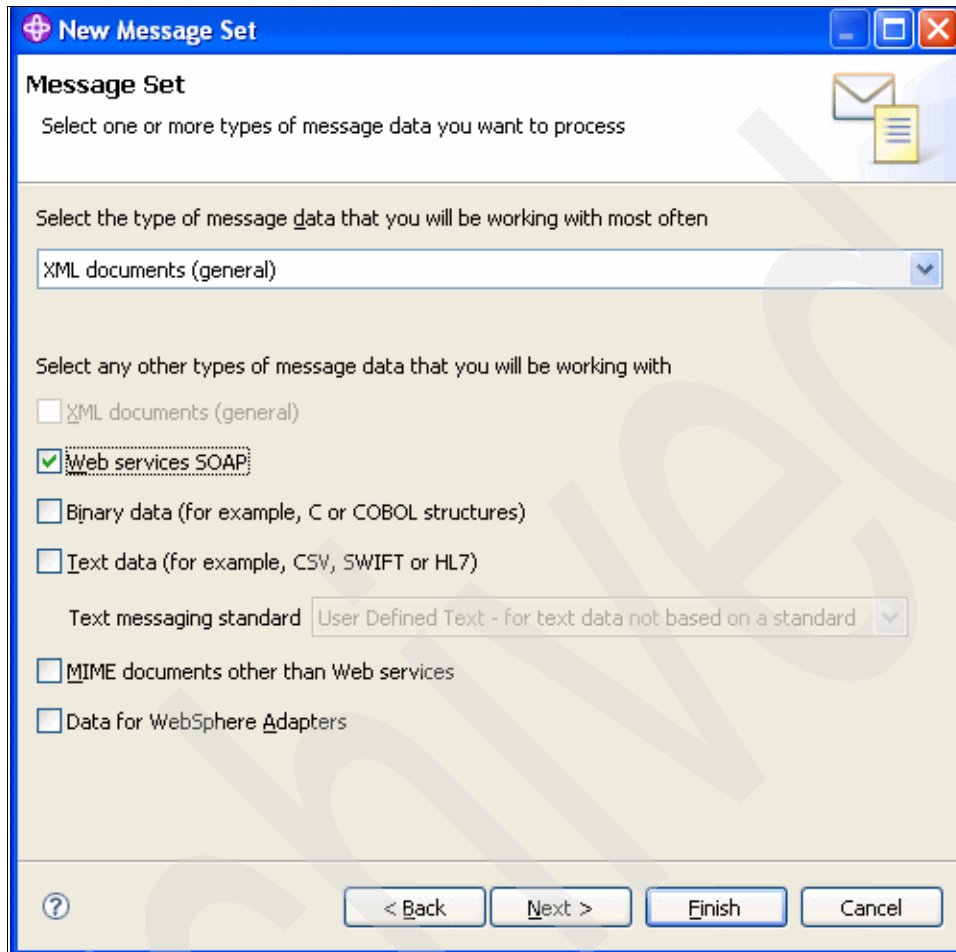


Figure 7-15 Message data options for the message set

- The starting point for the message set development is a Web Services Description Language (WSDL) file that defines the order status service. Refer to the WSDL file that is available in the setup files and the corresponding .xsd file, which defines the schema of the request and response messages. Use the option **New** → **Message Definition File** from → **WSDL file** and point to the WSDL file. See Figure 7-16.

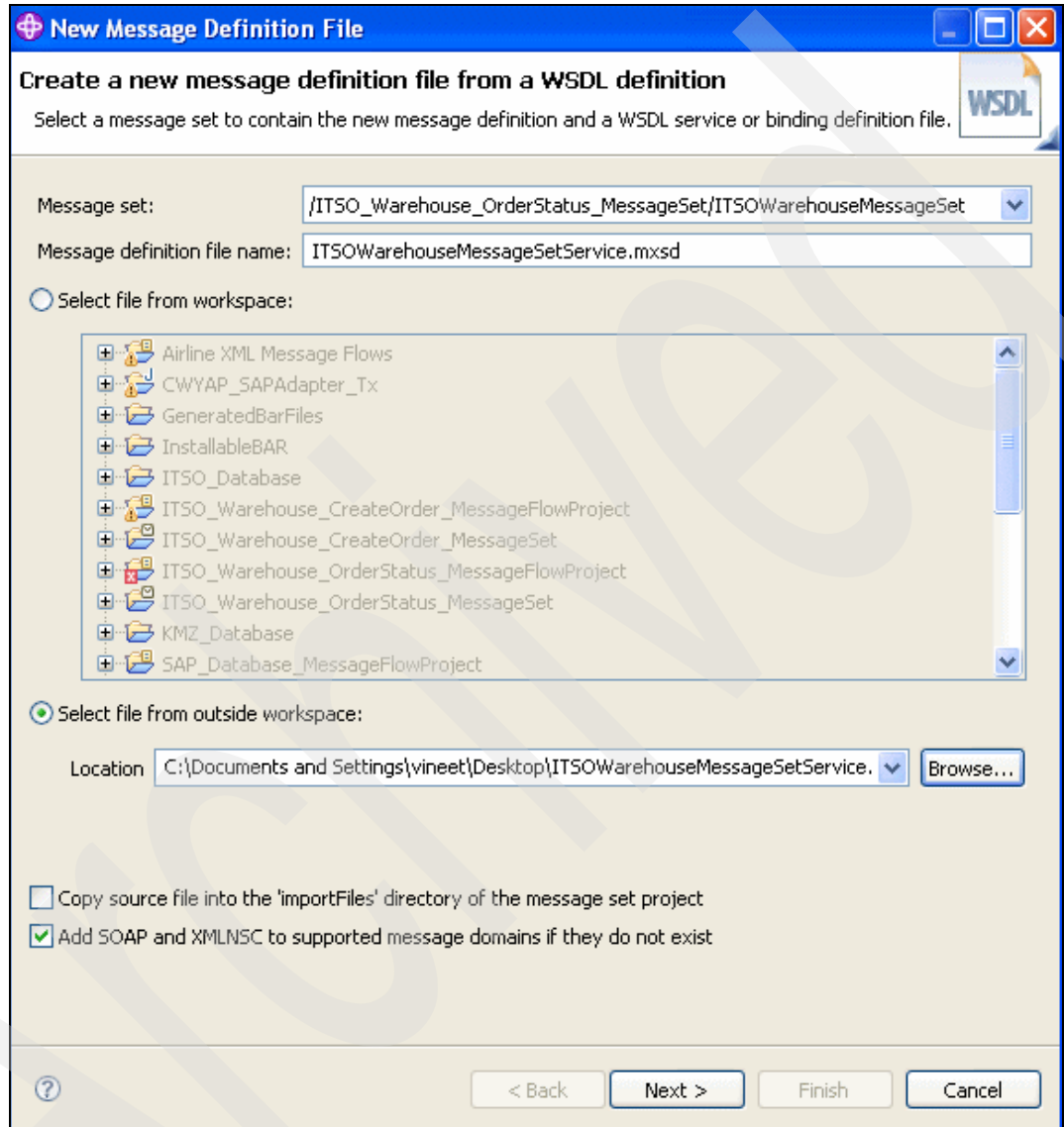


Figure 7-16 Importing the WSDL for the message definition

4. Click **Next** and then click **Finish** to import the wsdl file. Upon import, the message set project looks like Figure 7-17.

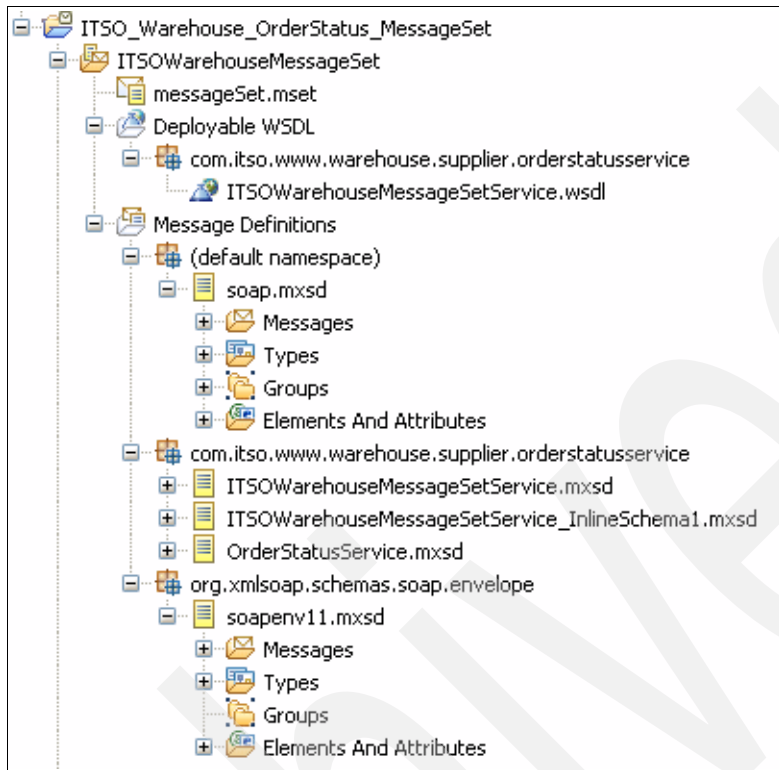


Figure 7-17 Message set project after the WSDL import

This step completes the message set development for the OrderStatus message flow.

ITSO_Warehouse_OrderStatus_MessageFlow

In this section, we develop the message flow that will be exposed as a Web service to the suppliers so that they can update the order status:

1. Create a new message flow ITSO_Warehouse_OrderStatus_MessageFlow within the project ITSO_Warehouse_OrderStatus_MessageFlowProject under the schema mqs i. Also, ensure that the message set project ITSO_Warehouse_OrderStatus_MessageSet is referenced by this new Message Flow project. See Figure 7-18.

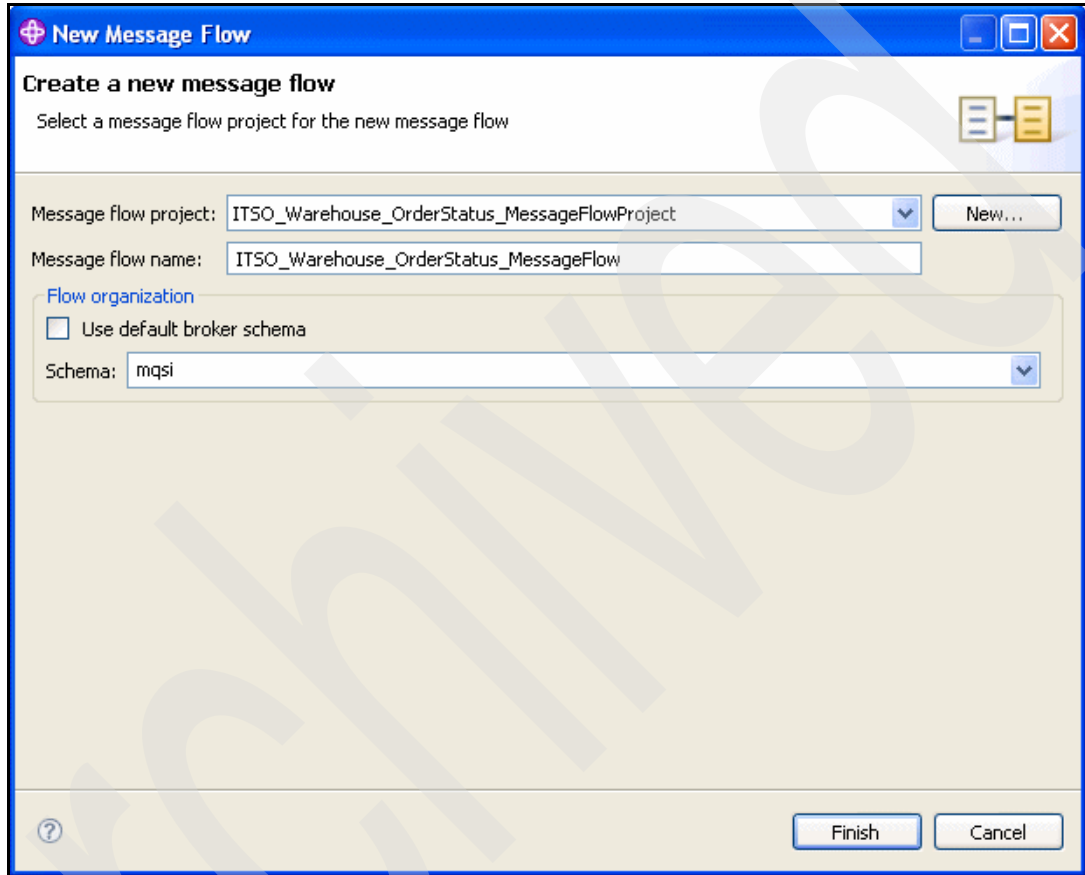


Figure 7-18 Creating a new message flow for the Order Status project

2. Create the message flow, as depicted in Figure 7-19. We will describe the configuration details for each node subsequently.

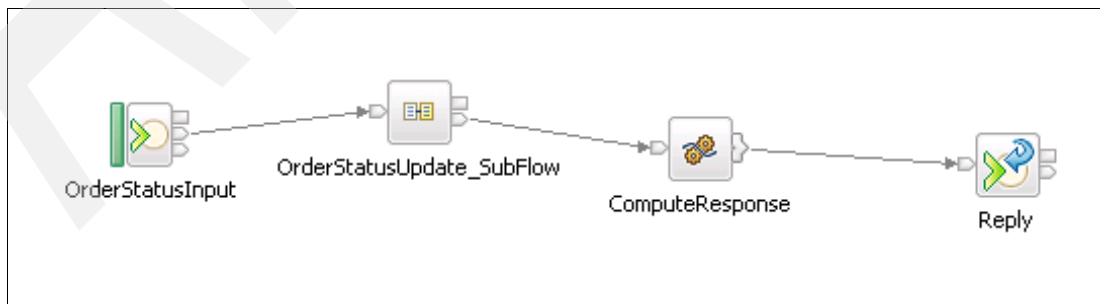


Figure 7-19 Message flow ITSO_Warehouse_OrderStatus_MessageFlow

- The OrderStatusInput SOAP Input node will be based on the WSDL file that we imported in the earlier section. Browse the project and select the WSDL file, as shown in Figure 7-20.

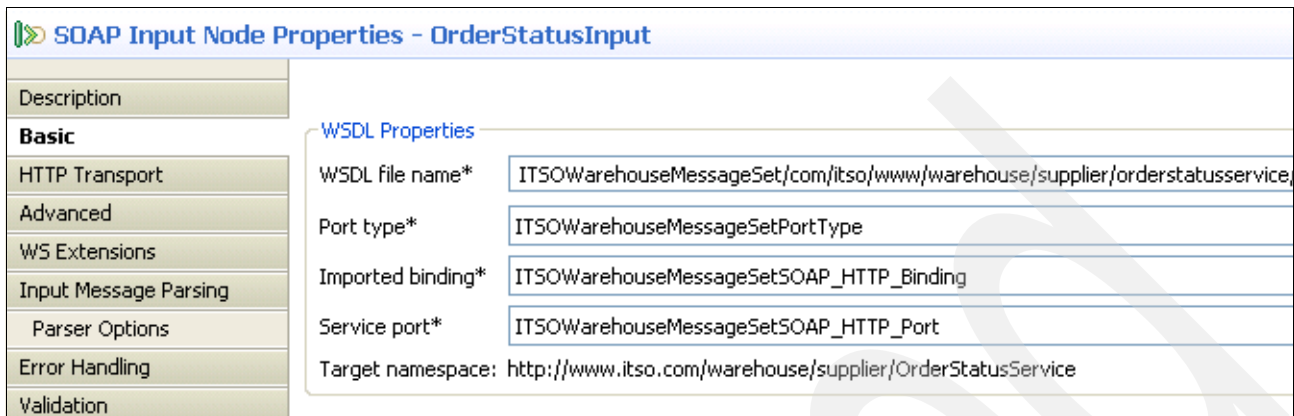


Figure 7-20 Properties configuration for OrderStatusInput node

- The ComputeResponse esql node is used to construct the return code for the web service. Example 7-11 shows the esql for the node. The flow ends with a SOAP reply node.

Example 7-11 esql for the node ComputeResponse esql node

```

BROKER SCHEMA mqsi
DECLARE ns NAMESPACE 'http://www.itso.com/warehouse/supplier/OrderStatusService';
CREATE COMPUTE MODULE OrderStatusUpdateFlow_Compute_Response
  CREATE FUNCTION Main() RETURNS BOOLEAN
  BEGIN

      SET OutputRoot.XMLNSC.ns:submitOrderStatusResponse.returnCode = 'SUCCESS';
      RETURN TRUE;

  END;
END MODULE;

```

- The subflow OrderStatusUpdate_SubFlow is used to extract the order ID and status of the order and to update the database. See Figure 7-21.

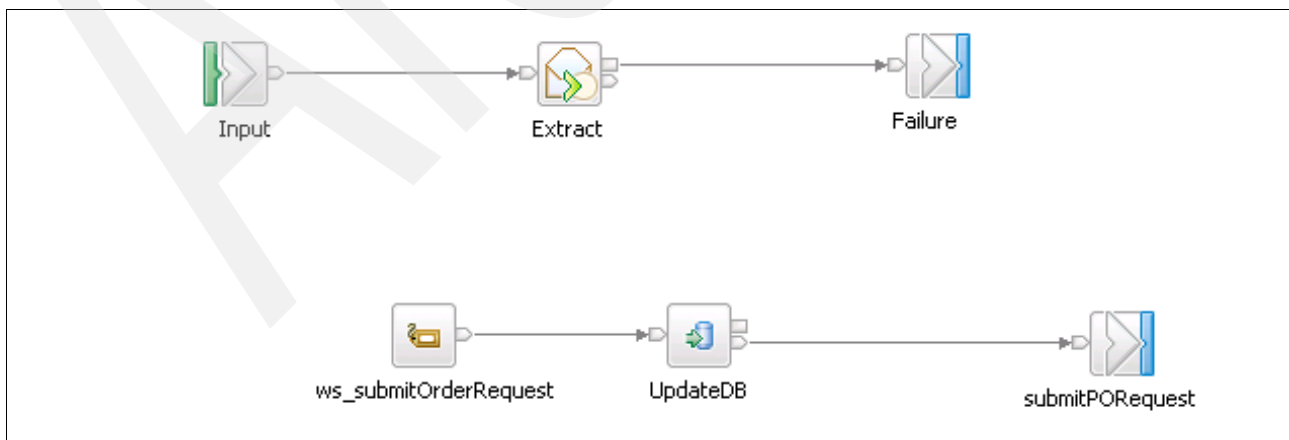


Figure 7-21 Message subflow OrderStatusUpdate_SubFlow

6. The Extract node is used to extract the order ID and status, as shown in Figure 7-22.

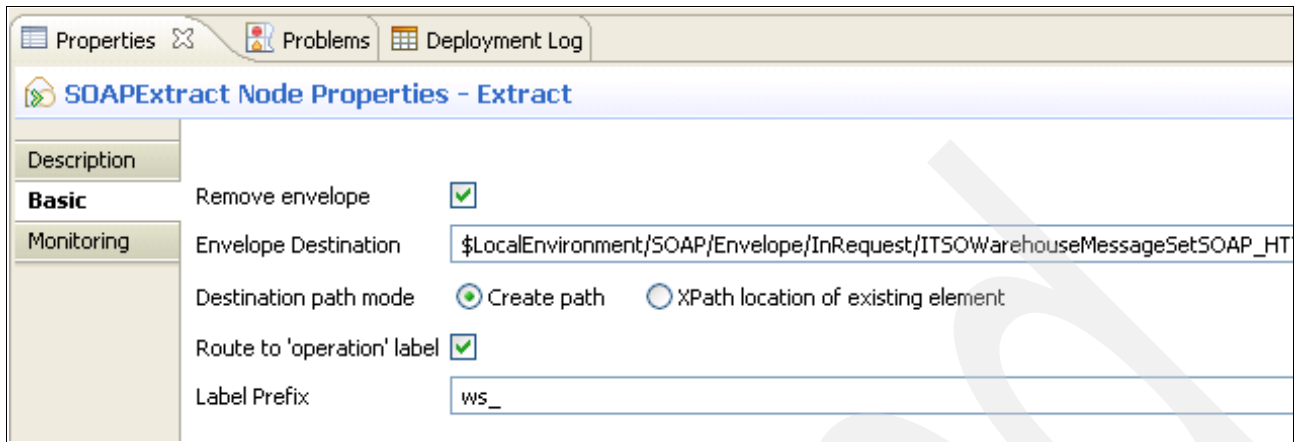


Figure 7-22 Configuration for the Extract node

- UpdateDB updates the database using the message map. Enter the Name OrderStatusRequest_Data_Insert to create the map. Enter the Schema mqsi. See Figure 7-23.

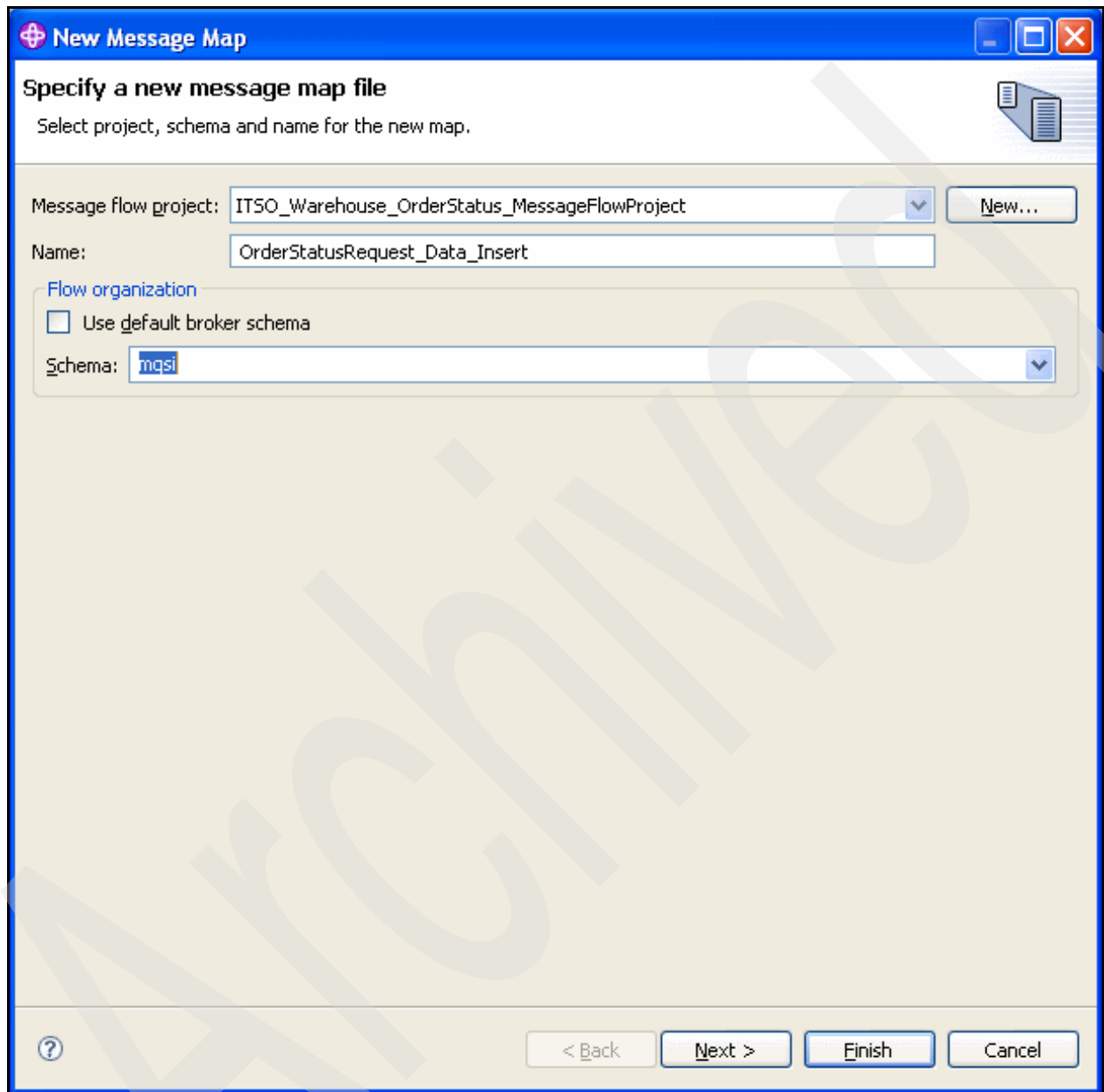


Figure 7-23 Creating the message map

- Click **Next** to specify the source and target for the map.

- For Select Map Sources, select **Messages** → **submitOrderRequest** and for Select Map Targets, select **Data Targets** → **Table Inserts** → **ORDER_STATUS**. See Figure 7-24.

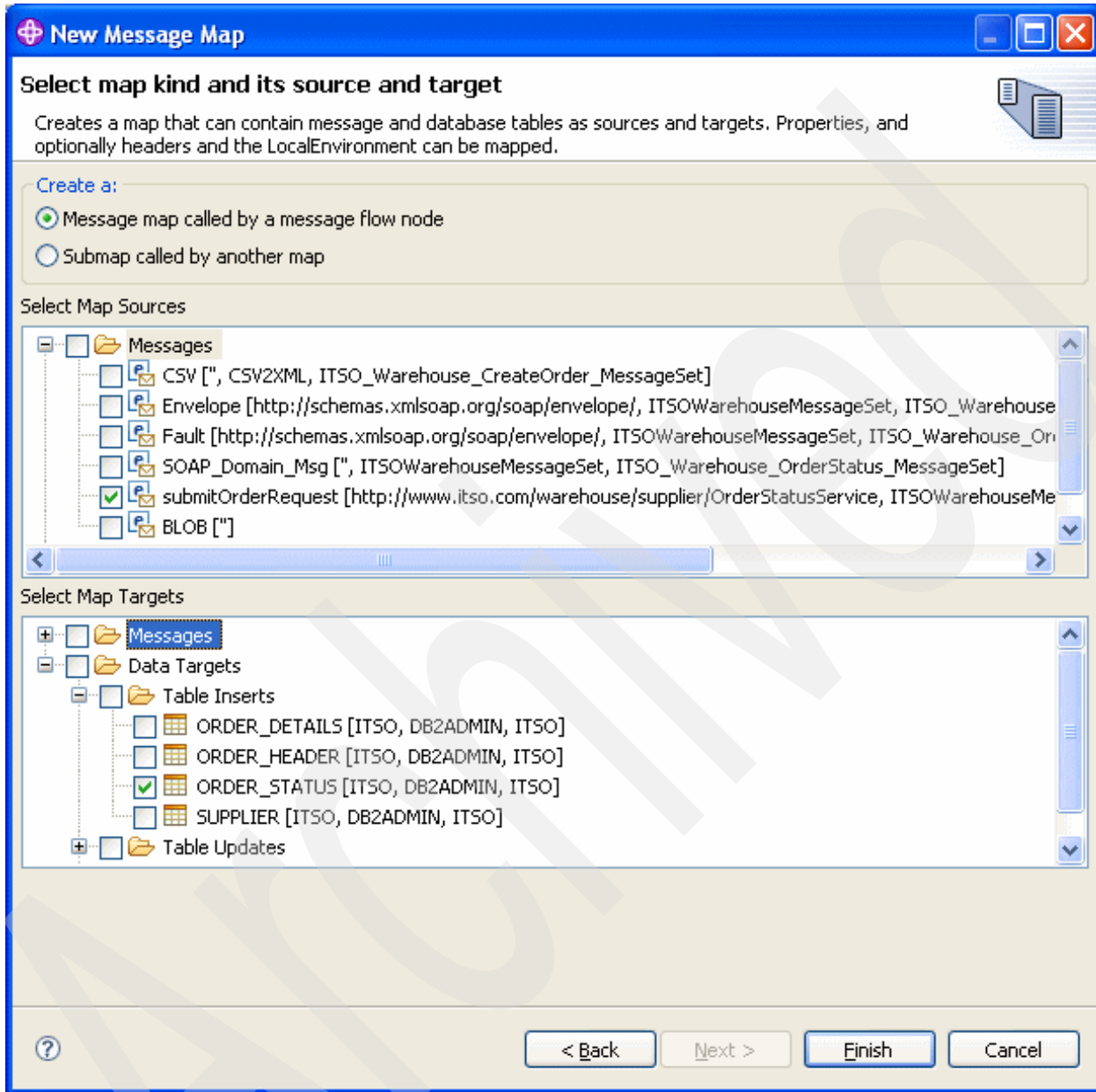


Figure 7-24 Specifying the source and target for the message map

- Click **Finish** to create the message map.

11. The mapping editor is displayed. Map the **orderId** and **status** from the **submitOrderRequest** to the **ORDERID** and **STATUS** columns of the table **ORDER_STATUS**, as shown in Figure 7-25.

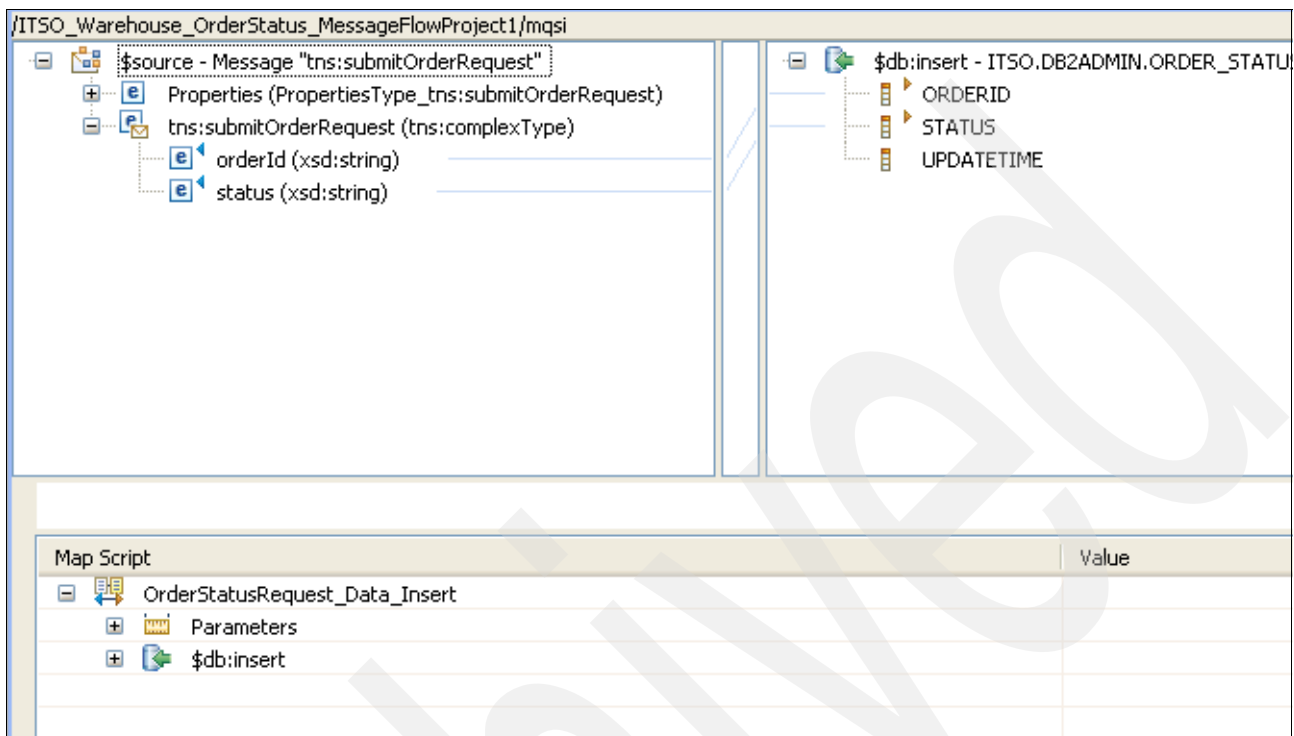


Figure 7-25 Mapping editor for the message map

12. The UpdateDB DataInsert node uses the message map **OrderStatusRequest_Data_Insert** to insert data in the **ORDER_STATUS** table. See Figure 7-26.

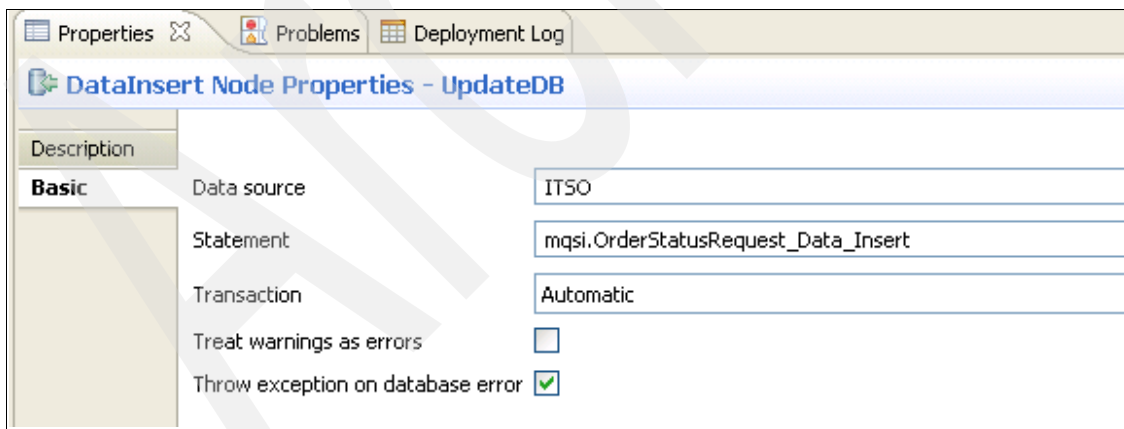


Figure 7-26 Configuration for UpdateDB Data Insert node

This step completes the flow development.

7.4.5 Testing the message flows

In this section, we describe how to test the message flows that we have developed as part of this solution.

Testing the create order flow

You can test the order creation flow for the JMS endpoints by copying the sample InvITSO output files 90045.csv and 90019.csv (provided in the setup files) to the directory C:\1z01\orders\suppliers\ (as specified on the Patterns Properties page, which is reflected in the FileInput node of ITSO_Warehouse_CreateOrder_MessageFlow).

Upon the successful run, there will be messages on the queues ITSO.JMS.90045.QUEUE (Figure 7-27) and ITSO.EDI.90019.QUEUE (Figure 7-28 on page 351).

Use `rfhuti1` (IH03: WebSphere Message Broker V7-Message display, test, and performance utilities), which is available at <http://www-01.ibm.com/support/docview.wss?uid=swg24000637> to view the messages. See Figure 7-27.

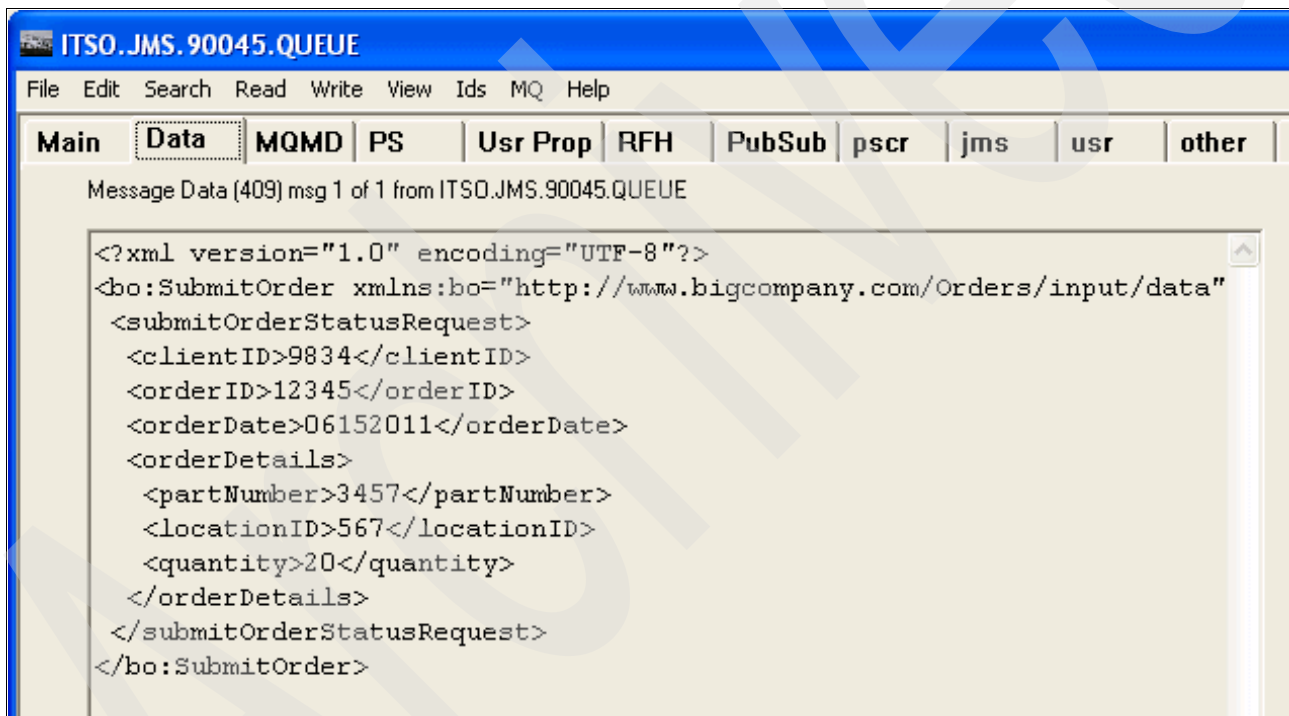


Figure 7-27 ITSO.JMS.90045.QUEUE

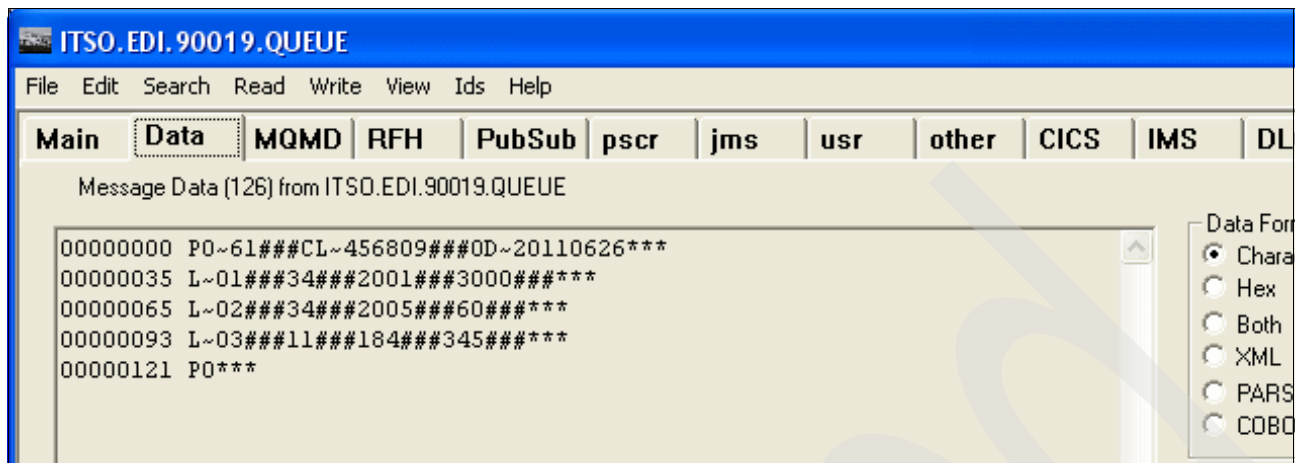


Figure 7-28 ITSO3 EDI order message

To test the Web service flow, deploy to the broker the sample flow, `Acme_CreateOrder_MessageFlow`, which is provided in the setup files. This flow mimics the Supplier Order Web Service for the supplier ITSO1.

Copy the sample order file `2344.csv` to the directory `C:\1z01\orders\suppliers\`. Also ensure that the endpoints that are specified in the database script `inserts.sql` are updated to the correct value. Remember that we had earlier used the command:

```
insert into SUPPLIER (supplierid, suppliername, forwardxsl, endpoint) values
(2344, 'ITS01', 'ITS01_Order.xsl', 'http://www.itso3.com:7080/Orders/web')
```

Upon a successful Web service call, success messages will be available in the queue `ITSO.WAREHOUSE.ORDERS.SOAP.RESPQ`. To confirm, we can also check the ORDER tables `ORDER_HEADER`, `ORDER_REQUEST`, and `ORDER_STATUS`.

Testing the order status flow

To test the order status flow, we need to mimic the supplier by developing a client for the Order Status web service. To do so, we can use the Web Services explorer feature in the Rational Application Developer product. This feature allows you to import the WSDL file, and it automatically creates the test client for the web service. The IBM developerWorks® article that is available at this website provides step-by-step guidance about how to use Web Services explorer:

<http://www.ibm.com/developerworks/webservices/tutorials/ws-eclipse-javase1/section7.html>

Import the Order Status WSDL from the setup files to Rational Application Developer and use the Web service explorer to send updates to the warehouse. To verify that the order status is updated, confirm by checking the table `ORDER_STATUS`.

7.5 Summary

In this chapter, we saw how we used the Evolve to service-oriented architecture (SOA) pattern to resolve the challenges being faced by the ITSO Enterprise warehouse. We also introduced the IBM product WebSphere Message Broker and showed its strengths, such as pattern-based development, message transformation, and support for multiple protocols. This

product helped ITSO Enterprise to reduce the impact of the changes that are made to the format and location of services, both in terms of impact to the applications and in terms of system management.

We also demonstrated the step-by-step implementation of the message flows using the Message Broker Toolkit.

Archived

Hybrid Bus pattern

This section introduces a Hybrid Bus scenario that improves and extends the Chapter 7, “Evolve to SOA pattern” on page 317 scenarios to secure the external partners’ interaction with ITSO Enterprise.

This chapter describes the following two business scenarios:

- ▶ Expose order web service
- ▶ Secure order fulfillment and order status messages

We cover the following topics in this chapter:

- ▶ 8.1, “Introduction to the business scenario” on page 354
- ▶ 8.2, “Architectural diagram” on page 354
- ▶ 8.3, “Benefits” on page 355
- ▶ 8.4, “Technical implementation” on page 355
- ▶ 8.5, “Summary” on page 394

8.1 Introduction to the business scenario

Expanding on the scenario that we described in Chapter 7, “Evolve to SOA pattern” on page 317, ITSO Enterprise decided to expose the Ordering Web service to business partners and customers to extend their business. However, the IT security department enforces certain security policies:

- ▶ All incoming requests must be authenticated. (Although a customer might choose a separate authentication mechanism, the Web Service-Security (WS-Security) username token is the most commonly used method.)
- ▶ All outgoing order requests must be secured.
- ▶ The transport layer must be protected.
- ▶ Sensitive information must be encrypted at the message level.

In addition, when the warehouse sends supplier-specific work orders, both the supplier orders and their status messages must be secured.

8.2 Architectural diagram

The Hybrid Bus architecture is deployed for implementing the security needs of the organization. The architecture extends WebSphere Enterprise Service Bus (ESB) and WebSphere Message Broker with the IBM WebSphere DataPower Integration Appliance XI50 to offer an end-to-end solution that meets both the functional and security requirements of the business.

DataPower is an ESB appliance that is positioned as an XML Security Gateway in the overall architecture. All interaction with external partners and customers is secured through DataPower by defining services in DataPower that implement two patterns: Web Service Proxy and Multi-protocol Gateway.

8.2.1 Scenario 1: Expose Order Web Service

The Web Service Proxy acts as an intermediary between external users and ITSO Enterprise’s Web Service. It directs the messages appropriately based on the processing policies that are defined in the proxy service (Figure 8-1).

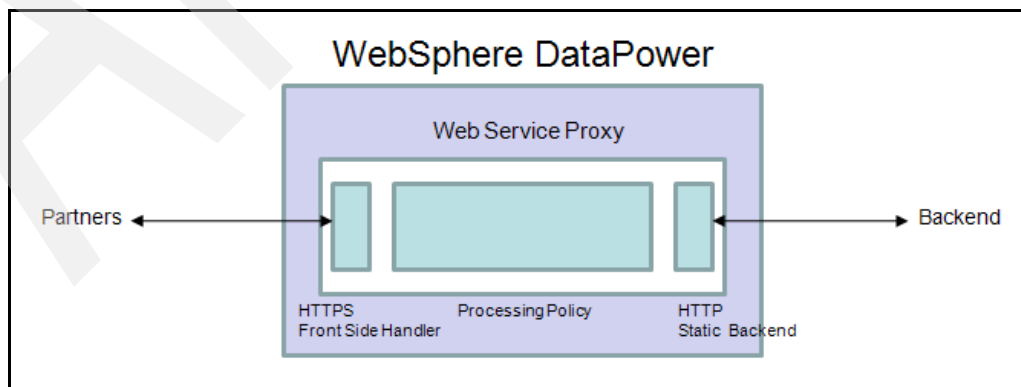


Figure 8-1 WebSphere DataPower

8.2.2 Scenario 2: Secure order fulfillment and order status messages

The Multi-protocol Gateway processes nearly any type of message, including SOAP, non-SOAP XML, text, or binary, and supports MQ and Java Message Service (JMS) transports in addition to HTTP. See Figure 8-2.

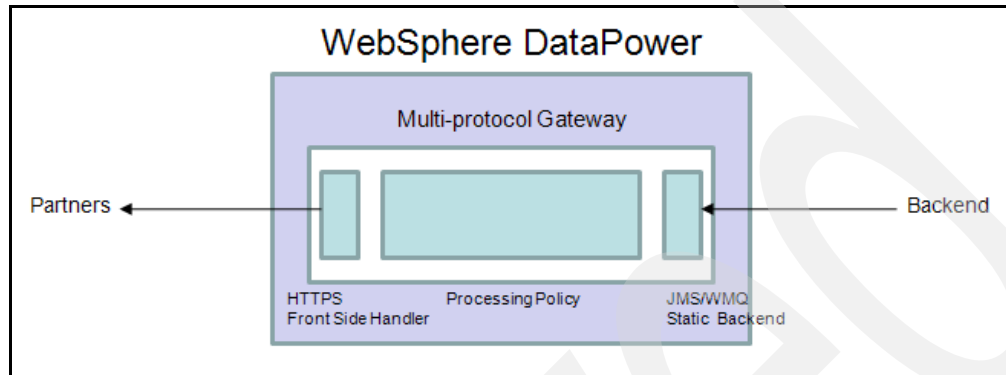


Figure 8-2 WebSphere DataPower

8.3 Benefits

Securing partners' and customers' interactions with ITSO Enterprise offers many benefits:

- ▶ Protection from data tampering
- ▶ Authentication and authorization
- ▶ Data encryption
- ▶ Non-repudiation

8.4 Technical implementation

In this section, we cover the technical implementation of the Hybrid Bus scenarios.

8.4.1 Ordering Web Service Proxy

The Web Service Proxy (WS-Proxy) provides a "Living" virtual service that passes messages between the client and the real service, so that the client actually connects directly to the proxy and not the back-end service. The importance of this approach is that the service consumer and the service provider do not need to be tightly coupled and hard-bound to one another. The details of accessing the service can be hidden from the consumer with DataPower.

In this scenario, we configure a WS-Proxy for the ordering process to off-load the security gateway functionalities from WebSphere ESB. In this scenario, we create a WS-Proxy Service that intercepts the Web service calls to a back-end queue. The proxy has the following characteristics:

- ▶ HTTPS protocol is used to secure the communication between the external customers and DataPower. DataPower is configured to terminate the Secure Sockets Layer (SSL) using a HTTPS Front-side Handler.

- ▶ The request SOAP message is authenticated via Lightweight Directory Access Protocol (LDAP) based on the identity that is extracted from the SOAP header.
- ▶ The request is routed to separate back ends based on the total amount of the order, which is calculated by the values within the message content.
- ▶ DataPower switches from HTTPS to MQ protocol to connect to the back-end queue.

Figure 8-3 shows the end-to-end, round-trip processing sequence for the inbound and outbound traffic that includes the requesting application, DataPower, WebSphere MQ, WebSphere ESB, and the .NET application.

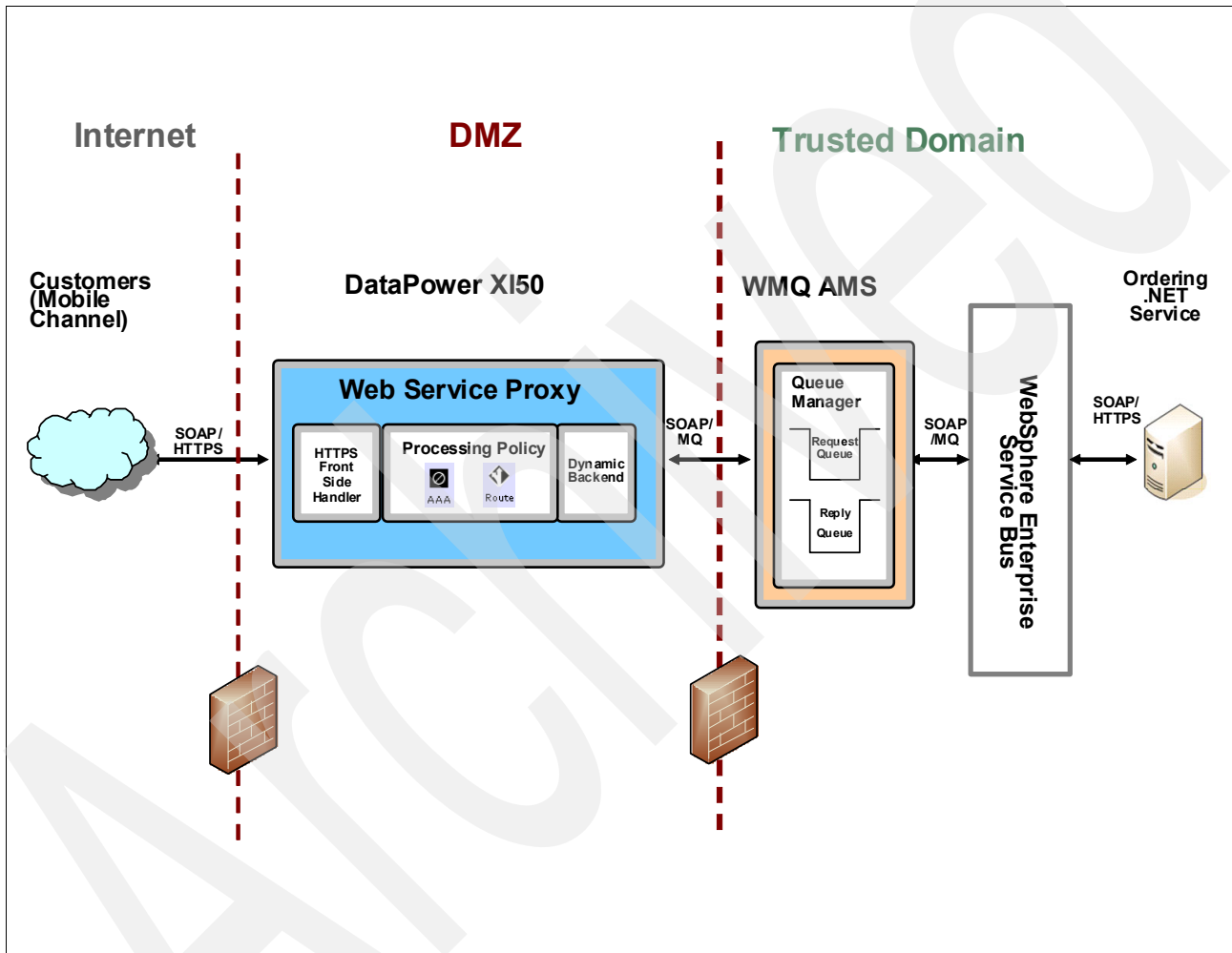


Figure 8-3 End-to-end processing

Creating the Web Service Proxy

The WS-Proxy object fully understands the Web Services Description Language (WSDL). In fact, a fully functional WS-Proxy can be configured simply by providing it with a WSDL document. After the WS-Proxy is configured, you can configure additional processing requirements and quality of service (QoS) parameters easily by using the WEBGUI.

To create a new WS-Proxy, follow these steps:

1. Open a Web Browser and enter `https://DataPowerAddress:9090/` into the address bar of the web browser. The DataPower WebGUI login page opens.

2. Log in to the DataPower WebGUI. The Control Panel appears.
3. Click the **Web Service Proxy** icon. It is located on the top row of service icons, as shown in Figure 8-4.

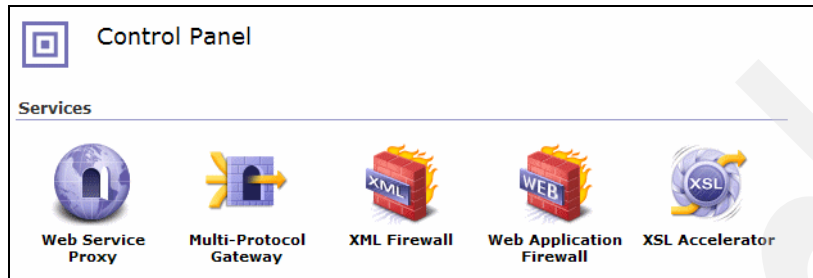


Figure 8-4 Web Service Proxy

4. Click **Add** to create a new **Web Service Proxy** object.
5. In the Web Service Proxy Name field, type `OrdersWSP` and click **Create Web Service Proxy**, as shown in Figure 8-5.

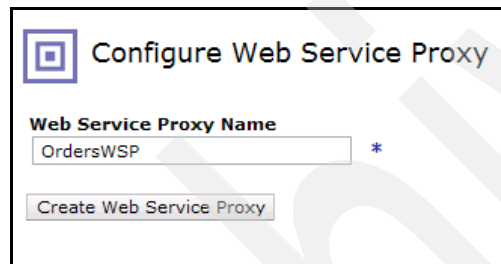


Figure 8-5 Creating a new Web Service Proxy

- The WSDL files tab of the Configure Web Service Proxy page appears, as shown in Figure 8-6. Click **Upload** in the WSDL File URL section.

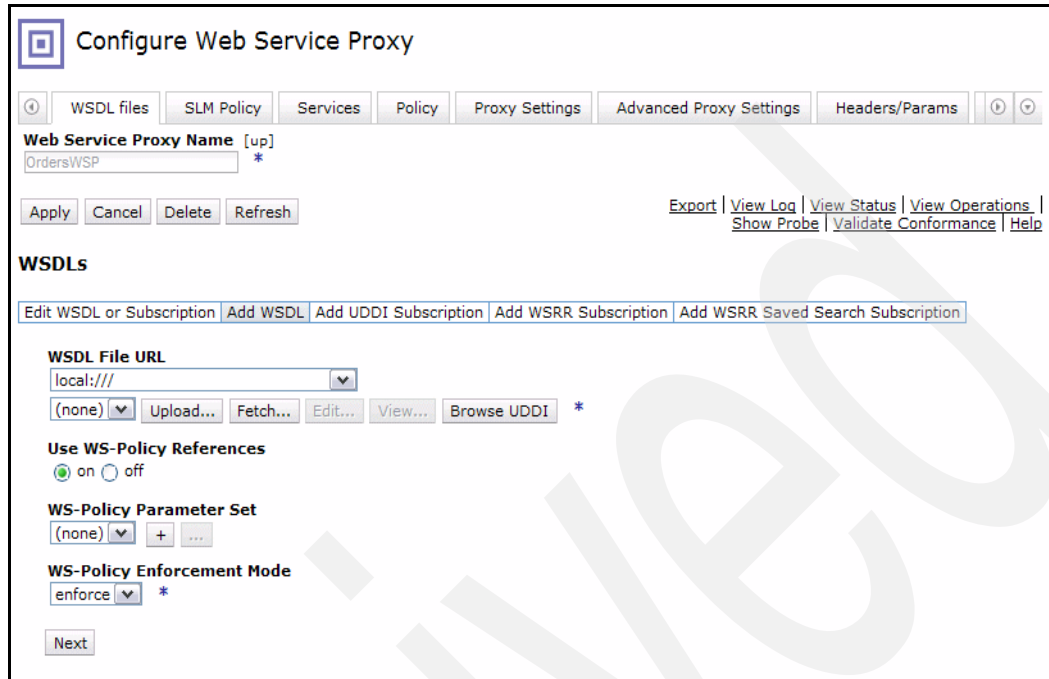


Figure 8-6 Adding WSDL to the Web Service Proxy

- Click **Browse** and open **ITSOOrdering.wsdl** and click **Upload** to begin the upload process, as shown in Figure 8-7.



Figure 8-7 Uploading the WSDL file

- Click **Next** at the bottom of the page.
- Create a new front-side handler to accept HTTPS requests for the WS-Proxy. Use the following procedure to configure the HTTPS Front-Side Handler:
 - In the Local Endpoint Handler field, click the plus sign (+) to create a new endpoint handler. A pop-up menu appears that lists the possible front-side handlers.

Front-Side Handler: A front-side handler is the entry point to several services. A handler listens on a specific IP address-port for incoming requests (or polls for messages) and performs a certain level of validation on the requests. Most validation is protocol-specific.

- b. In the pop-up menu, click **HTTPS (SSL) Front Side Handler**, as shown in Figure 8-8. A new window opens, displaying the properties of the HTTPS Front Side Handler.

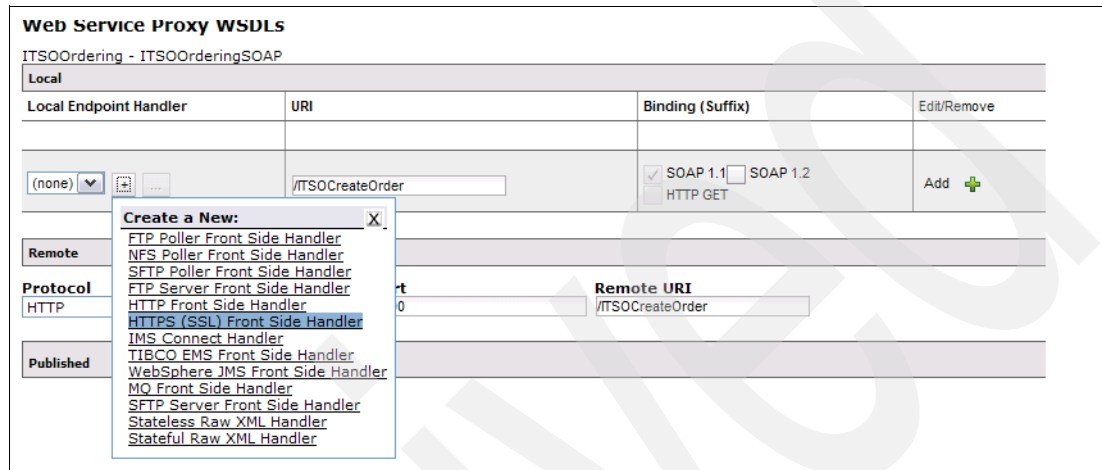


Figure 8-8 Adding local endpoint handler

- c. Configure the new **HTTPS Front Side Handler** with the following values, as shown in Figure 8-9.

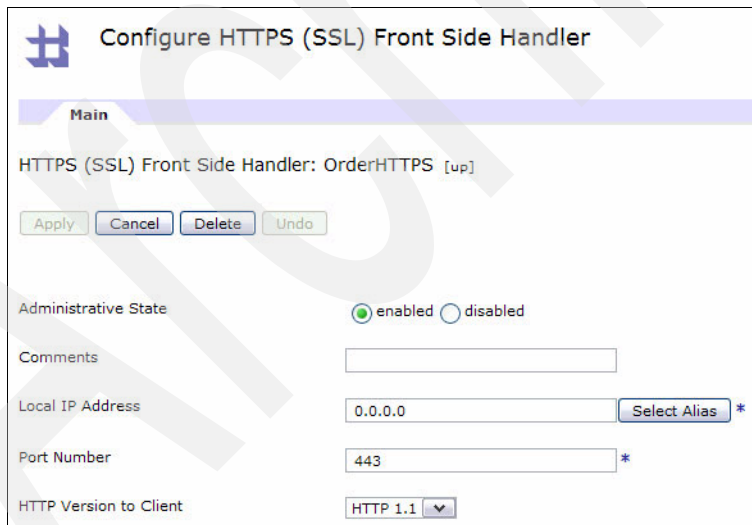


Figure 8-9 Configuring HTTPS (SSL) Front Side Handler

Local IP Address: The default is 0.0.0.0, which indicates that the service is active on all IP addresses that are assigned to DataPower. You can limit the traffic for the WS-Proxy to a specific port on DataPower.

Advanced configuration of the HTTPS Front Side Handler: You can filter the incoming requests based on the HTTP protocol-level properties, as shown in Figure 8-10 on page 360. The following address explains the detailed configuration parameters:

http://publib.boulder.ibm.com/infocenter/wsdatap/v3r8m2/topic/xi50/https-handler_configuring_task.htm#https-handler_configuring_task

The screenshot shows the following configuration options:

- Allowed Methods and Versions:**
 - HTTP 1.0
 - HTTP 1.1
 - POST method
 - GET method
 - PUT method
 - HEAD method
 - OPTIONS
 - TRACE method
 - DELETE method
 - URL with Query Strings
 - URL with Fragment Identifiers
 - URL with ..
 - URL with cmd.exe
- Persistent Connections:** on off
- Compression:** on off
- Maximum Allowed URL Length:** 16384
- Maximum Allowed Total Header Length:** 128000
- Maximum Number of HTTP Request Headers Allowed:** 0
- Maximum Allowed Length of HTTP Header Name:** 0
- Maximum Allowed Length of HTTP Header Value:** 0
- Maximum Allowed Length of HTTP Query String:** 0

Figure 8-10 Configuring the advanced filtering parameters

- d. Select the instance of the SSL Proxy profile object to assign from the SSL Proxy list box, as shown in Figure 8-11. The procedure to create the SSL proxy is described in “SSL termination” on page 361.

The screenshot shows the following configuration:

- SSL Proxy:** OrdersSSLProfile
- Access Control List:** (none)

Figure 8-11 Assigning the SSL Proxy

- e. Click **Apply** in the upper-left corner of the Configure HTTPS Front Side Handler page.

10. Click **Add** to the right of the **Local Endpoint Handler** information, as shown in Figure 8-12.

Figure 8-12 Adding the local endpoint handler

11. Click **Apply** to save the new WS-Proxy service. You will then see that the service is up and running, as shown in Figure 8-13.

Figure 8-13 WS-Proxy status

SSL termination

The connections to the DataPower appliance are required to be SSL terminated. Data that is transmitted over SSL connections is encrypted using session keys that are secured using public key cryptography. *Public key cryptography* requires a public key (which is stored in a certificate) and a private key. The use of key pairs (public/private) is known as *asymmetric* encryption. It is vital that the private key is protected, while its public counterpart, the public key (which is often carried in a certificate), can be freely distributed. Certificates are typically validated by a *Certificate Authority (CA)*. In the event that an authority needs to revoke a previously distributed certificate, it adds the revoked certificate to a globally published *certificate revocation list (CRL)*.

To enable SSL, you need to create the following objects in DataPower:

- ▶ Crypto profile:
 - Crypto identification credentials
 - Crypto validation credentials
 - Crypto key
 - Crypto certificate
- ▶ SSL proxy profile

The following sections describe the required steps to generate keys and certificates, and how to use them to support SSL.

Generating the public and private key pair

DataPower provides powerful tools to generate public and private keys. For this scenario, we use these tools to create a key and a self-signed certificate. Note that in a production or actual client environment, the certificate is signed by a trusted certificate authority (CA). In

addition to the capability of generating public and private keys, DataPower also supports uploading files from a Java keystore (JKS).

To create the certificate and private key, perform the following steps:

1. Using the DataPower Web GUI, navigate to the cryptographic tools page by selecting **Administration** → **Crypto Tools**.
2. The Generate Key page appears. Enter the details that are shown in Figure 8-14.

The screenshot shows the 'Crypto Tools' web interface. The left sidebar contains a navigation tree with categories: Status, Services, Network, Administration, Main, Configuration, Access, Device, Debug, and Miscellaneous. The main area is titled 'Crypto Tools' and has a sub-header 'Generate Key'. Below this are several tabs: 'Generate Key' (selected), 'Export Crypto Object', 'Import Crypto Object', and 'Add SSH Known Host'. A 'Help' link is in the top right. The form fields are as follows: 'LDAP (reverse) Order of RDNs' with radio buttons for 'on' and 'off' (off selected); 'Country Name (C)' with a text box containing 'US'; 'State or Province (ST)' with a text box containing 'NC'; 'Locality (L)' with an empty text box; 'Organization (O)' with a dropdown menu showing 'ITSOEnterprise'; 'Organizational Unit (OU)' with a dropdown menu showing 'Orders Management'; 'Organizational Unit 2 (OU)', 'Organizational Unit 3 (OU)', and 'Organizational Unit 4 (OU)' with empty text boxes; 'Common Name (CN)' with a text box containing 'orderskey' and an asterisk; 'RSA Key Length' with a dropdown menu showing '1024 bits'; 'File Name' with a text box containing 'orderskey'; 'Validity Period' with a text box containing '365' and 'days' next to it; 'Password' with a masked text box and a 'Confirm Password' field with a masked text box; 'Password Alias' with an empty text box; 'Export Private Key' with radio buttons for 'on' and 'off' (off selected); 'Generate Self-Signed Certificate' with radio buttons for 'on' and 'off' (on selected); 'Export Self-Signed Certificate' with radio buttons for 'on' and 'off' (on selected); 'Generate Key and Certificate Objects' with radio buttons for 'on' and 'off' (on selected); 'Object Name' with a text box containing 'orderskey'; and 'Using Existing Key Object' with an empty text box. A 'Generate Key' button is located at the bottom of the form.

Figure 8-14 Generating a private key and self-signed certificate using Crypto Tools

3. Click **Confirm** to create the key and certificate. The appliance generates the following objects:
 - Private key on the hardware security module (HSM) and a copy in temporary:///orderskey-privkey.pem
 - Certificate signing request in temporary:///orderskey.csr
 - Self-signed certificate in temporary:///orderskey.csr

- Crypto key object named orderskey and a crypto certificate object named orderskey

Key Files: All key files are placed in an encrypted storage area on the appliance; the appliance can read them, but the values cannot be displayed to users. You can export private keys on *only HSM-equipped* DataPower appliances.

Figure 8-15 shows the key files that are generated in the file system.

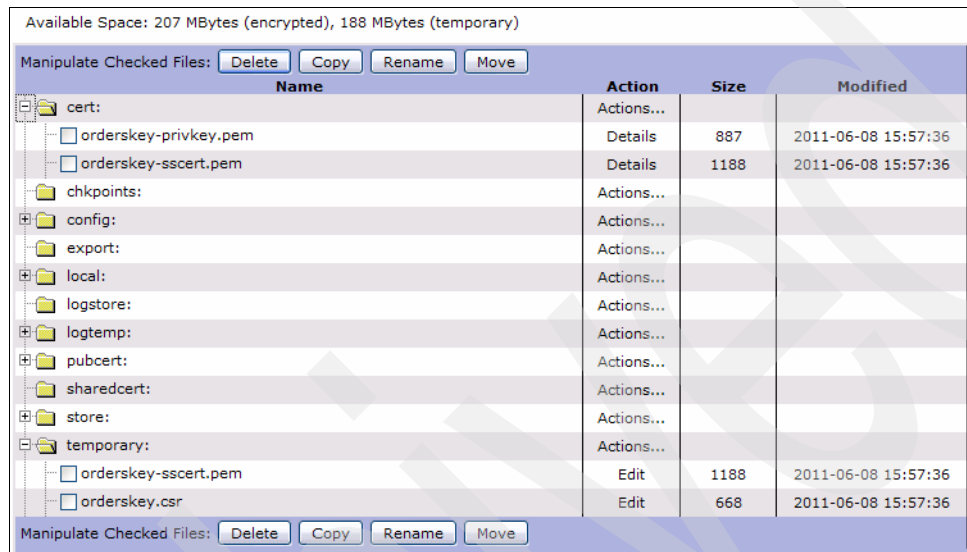


Figure 8-15 Keys in the file system

Creating a crypto identification credential

DataPower uses a *crypto identification credential* to associate or match a public key and private key for use in cryptographic operations, such as establishing SSL connections. We use the certificate and private key that were created in the previous section to build a new crypto identification credential.

To create a crypto identification credential, perform the following steps:

1. Click the **Control Panel** link to display the control panel.
2. Click the **Keys & Certs Management** icon. It is located on the bottom row of service icons, as shown in Figure 8-16.

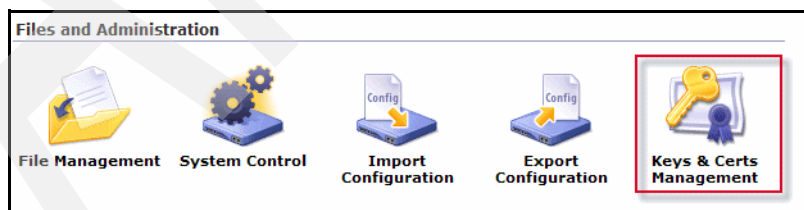


Figure 8-16 Keys and Certificate Management

- Click the **Identification Credentials** link on the Keys and Certificates Management page, as shown in Figure 8-17.



Figure 8-17 Keys and Certificate Management Page

- On the **Configure Crypto Identification Credentials** page, click **Add**.
- Complete the Configure the Crypto Identification Credential page by using the information that is shown in Figure 8-18.

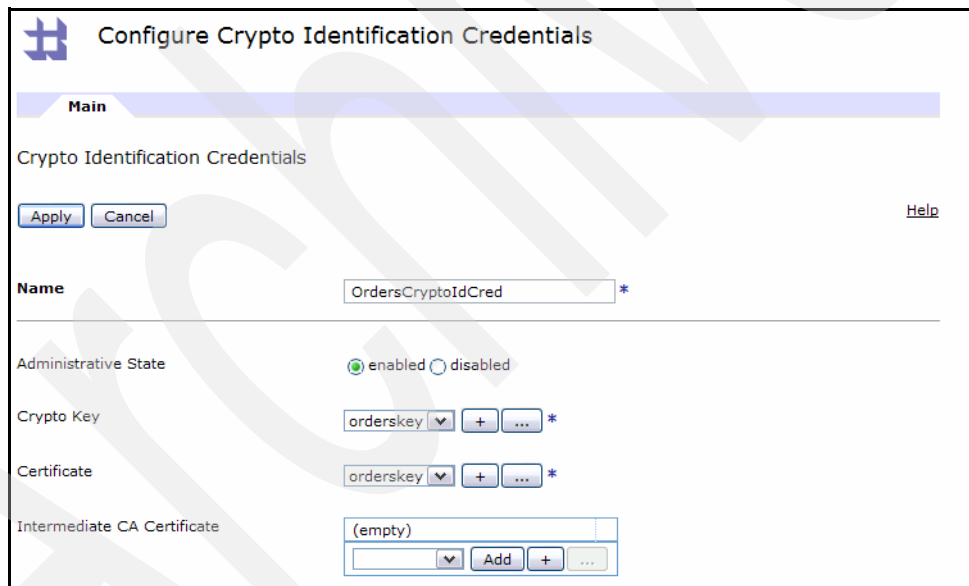


Figure 8-18 Configure Crypto Identification Credentials

- Click **Apply** to save the changes.

Creating a validation credential

When a client connects to a service using mutual SSL, the client is required to provide a certificate during the SSL handshake. If that certificate (or its root certificate) does not exist in the crypto validation credential, the handshaking will fail. In this step, you upload another certificate, which represents the partner that will connect to your service.

To create a crypto validation credential, perform the following steps:

1. Navigate to the Validation Credentials creation page by selecting **Objects** → **Crypto** → **Validation Credentials**. The Configure Crypto Validation Credentials page appears. (You can access this page through the Control Panel, as shown in Figure 8-17 on page 364.)
2. Click **Add** to configure a new validation credential. Configure the new validation credential, as shown in Figure 8-19.

The screenshot shows the 'Configure Crypto Validation Credentials' page. It features a 'Main' tab and a 'Crypto Validation Credentials' section. The 'Name' field is 'OrdersCryptoValCred'. The 'Administrative State' is 'enabled'. The 'Certificates' list contains 'orderskey'. The 'Certificate Validation Mode' is 'Match exact certificate or immediate issuer'. The 'Use CRL' option is 'on'. The 'Require CRL' option is 'off'. The 'CRL Distribution Points Handling' is 'Ignore'. There are 'Apply' and 'Cancel' buttons at the top left and a 'Help' link at the top right.

Figure 8-19 Configure Crypto Validation Credentials

3. Click **Apply** to save the changes.

Creating a crypto profile

A crypto profile in DataPower associates a crypto identification with a crypto validation credential. A crypto profile is required when configuring an SSL proxy profile.

To create a crypto profile, perform the following steps:

1. Navigate to **Objects** → **Crypto** → **Crypto Profile** in the DataPower Web GUI. The Configure Crypto Profile page is shown.

2. Click **Add** to create a new crypto profile, and enter the details, as shown in Figure 8-20.

Figure 8-20 Configure Crypto Profile

Validation Credentials: Because the validation credential object is unspecified, it implies that the DataPower appliance does not request a certificate from the client.

3. Click **Apply** to save the new crypto profile.

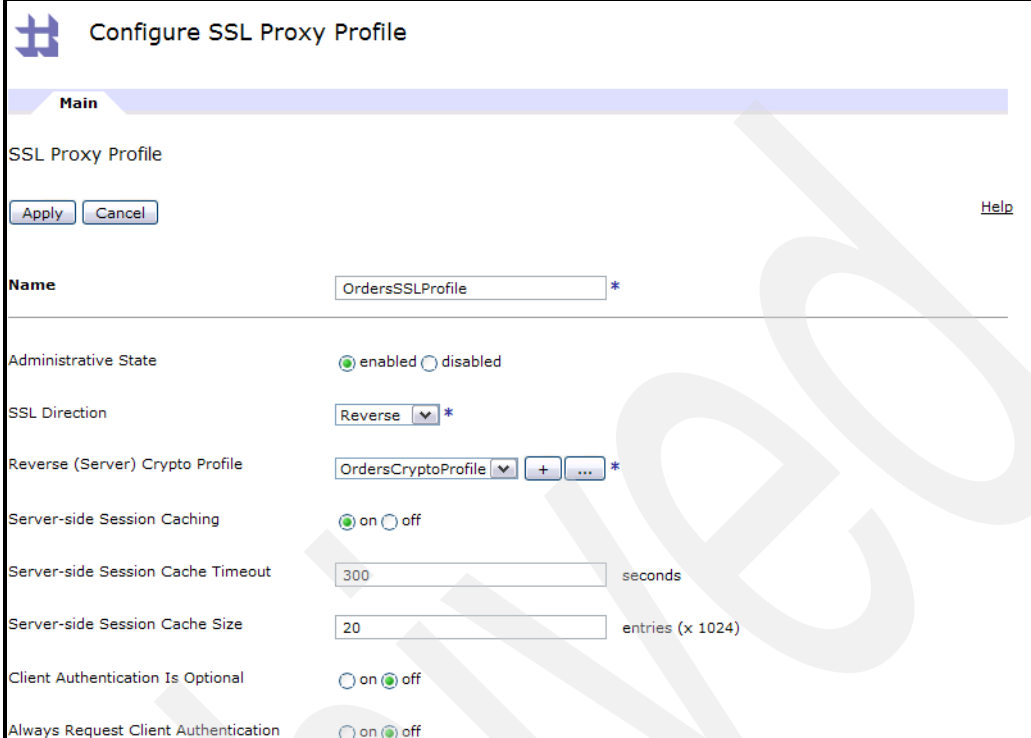
Creating an SSL proxy profile

The SSL proxy profile object provides the SSL-specific configuration parameters and references the crypto profile for any required crypto keys and certificates.

Create an SSL proxy profile by performing the following steps:

1. Navigate to **Objects** → **Crypto** → **SSL Proxy Profile** in the DataPower Web GUI. The Configure SSL Proxy Profile page appears.
2. Click **Add** to create a new SSL proxy profile.
3. In the SSL Direction drop-down list box, select **Reverse**. (DataPower is acting as the SSL server.)

4. In the Reverse (Server) Crypto Profile drop-down list box, select **OrdersCryptoProfile**. The configuration is shown in Figure 8-21.



The screenshot shows the 'Configure SSL Proxy Profile' configuration page. The 'Name' field is 'OrdersSSLProfile'. The 'Administrative State' is 'enabled'. The 'SSL Direction' is 'Reverse'. The 'Reverse (Server) Crypto Profile' dropdown is set to 'OrdersCryptoProfile'. The 'Server-side Session Caching' is 'on'. The 'Server-side Session Cache Timeout' is '300' seconds. The 'Server-side Session Cache Size' is '20' entries (x 1024). The 'Client Authentication Is Optional' is 'off'. The 'Always Request Client Authentication' is 'off'.

Figure 8-21 Configure SSL Proxy Profile

5. Click **Apply** to save the new SSL proxy profile.

Access control

The ITSO Enterprise IT Security Department has mandated that a proper authentication mechanism must be applied to Web Services. In this scenario, the user's identity will be verified using a Web Services Security (WS-Security) UsernameToken and authenticated against LDAP.

WS-Security Username Token: WS-Security is a suite of specifications that define the mechanisms for transporting a user identity in a Web service environment. The UsernameToken profile describes the transportation of a username and password in the header of a SOAP message. The password field of UsernameToken must be protected via token encryption or by the use of transport-level security (SSL).

DataPower provides a powerful and flexible Authentication, Authorization, and Audit (AAA) policy. The AAA policy, which is sometimes referred to as an *Access Control Policy*, identifies a set of resources and procedures that can be used to determine whether a requesting client is granted access to a specific service, file, or document. AAA policies can be considered a type of filter in that they accept or deny a specific client request.

Figure 8-22 on page 368 shows the processing for an AAA policy. The initial processing, which is common to all policies, consists of extracting the claimed identity of the service requester and the requested resource from an incoming message and its protocol envelope. As you define an AAA policy, the extraction methods are specified by a series of check boxes that enable one or more identity and resource extraction methods. A wide range of identity

and resource extraction methods is supported. For example, the identity can be based on the IP address, form (account name and password), Security Assertion Markup Language (SAML) assertion, or other criteria. The requested resource can be specified by an HTTP URL, a namespace, a WSDL method, or others.

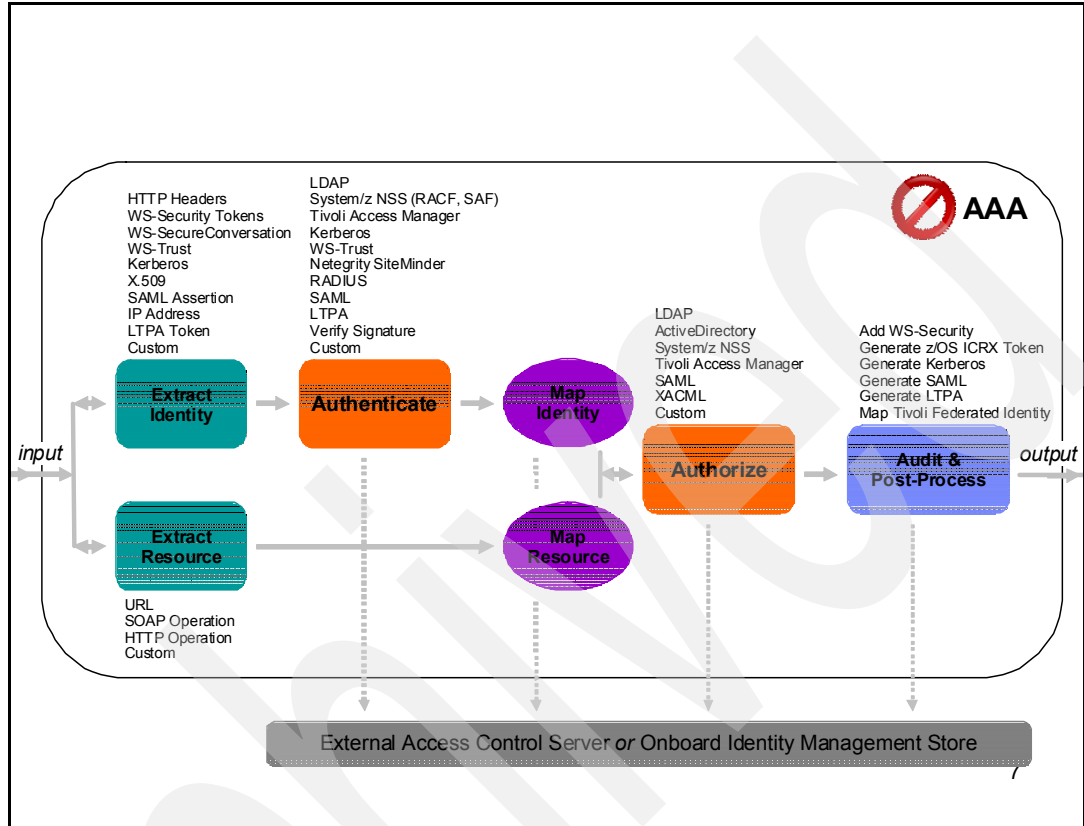


Figure 8-22 AAA framework

Configuring an AAA policy

To configure an AAA policy, perform the following steps:

1. Open the processing rule that is defined for the order creation operation by clicking **Processing Rules** in the WSDL Policy Tree Representation page, as shown in Figure 8-23 on page 369.

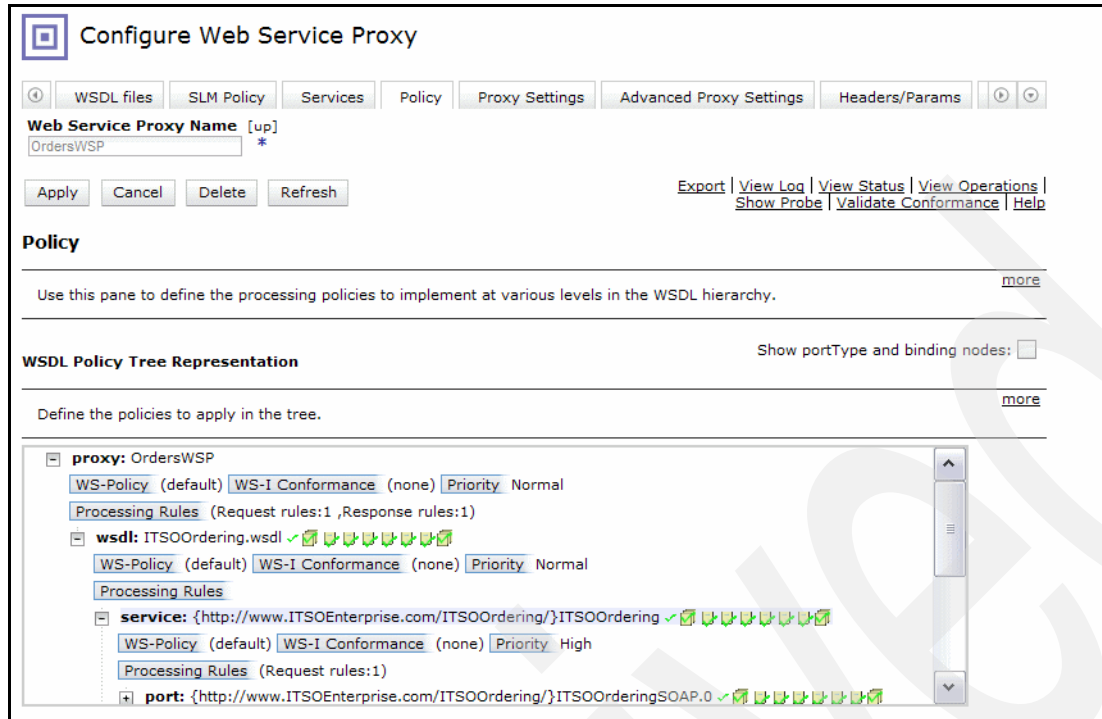


Figure 8-23 WSDL Policy Tree Representation

Policies at various levels: You can define policies at separate levels within the WSDL hierarchy:

- ▶ Global
- ▶ WSDL-specific
- ▶ Service-specific
- ▶ Port-specific
- ▶ Operation-specific

We define a service-specific policy for the ITSOOrdering service only.

2. Drag an **AAA** action icon and drop it after the initial Match action, as shown in Figure 8-24.

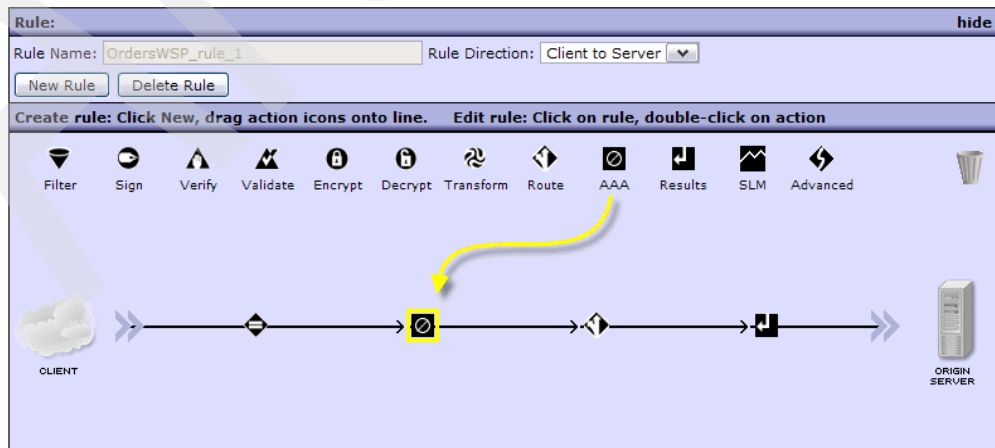


Figure 8-24 AAA action

3. Double-click the **AAA action** icon to open the Configure AAA Action dialog box.
4. Click the plus sign (+) beside the AAA Policy list to create a new AAA policy, as shown in Figure 8-25.

Figure 8-25 Configure an AAA Action

5. In the AAA Policy Name field, enter the name for the new AAA policy and click **Create** to start the interactive editor.
6. Select **Password-carrying UsernameToken Element from WS-Security Header** as the method to be used to extract a user's identity from an incoming request, as shown in Figure 8-26.

Figure 8-26 Define how to extract a user's identity from an incoming request

7. Select the **Bind to Specified LDAP Server** as the method to authenticate the user and configure the LDAP connection, as shown in Figure 8-27 on page 371.

Authentication cache: The *authentication cache* stores authentication data to minimize the overhead of re-authenticating the same identity.

IS0 | Configure: Web Service Proxy | **Configure an Access Control Policy** | [Help](#)

AAA Policy Name: OrdersAAAPolicy

Define how to authenticate the user.

Method

- Accept a SAML Assertion with a Valid Signature
- Accept an LTPA token
- Bind to Specified LDAP Server
- Contact a SAML Server for a SAML Authentication Statement
- Contact a WS-Trust Server for a WS-Trust Token
- Contact ClearTrust Server
- Contact Netegrity SiteMinder
- Contact NSS for SAF Authentication
- Contact Tivoli Access Manager
- Custom Template
- Pass Identity Token to the Authorize Step
- Retrieve SAML Assertions Corresponding to a SAML Browser Artifact
- Use an Established WS-SecureConversation Security Context
- Use certificate from BinarySecurityToken
- Use DataPower AAA Info File
- Use specified RADIUS Server
- Validate a Kerberos AP-REQ for the Correct Server Principal
- Validate the Signer Certificate for a Digitally Signed Message.
- Validate the SSL Certificate from the Connection Peer

*

LDAP Load Balancer Group (none) + ...

Host 9.12.5.223

Port 389

SSL Proxy Profile (none) + ...

LDAP Bind DN cn=root,dc=itsoenterprise,dc=com

Figure 8-27 Define how to authenticate the user

- In the Extract Resource form, select **Local Name of Request Element** and click **Next**.

Local name of the request element: Because the message is a SOAP request, you can expect that the first element in the SOAP body contains the operation that is being requested, which is known as the local name of the request element.

- For the authorization phase, leave the default set to **Allow Any Authenticated Client** and click **Next**.

Credential or resource mapping: We do not need to perform any credential or resource mapping in this scenario. Although this step is optional, it is important to note that it ensures the interoperability between systems (the IBM DataPower device® and the system in charge of authorization) in case it is not possible to use the extracted identity to authorize the use of the requested resource, especially when authorization is managed by an external authority.

10. On Figure 8-28, select **Run Custom Post Processing Stylesheet**. In the URL, select the folder as **store:///** and the file name as **strip-security-header.xsl**.

The screenshot shows the 'Configure an Access Control Policy' interface for 'OrdersAAAPolicy'. It is divided into sections: 'Monitors', 'Logging', and 'Choose any post processing.'. In the 'Choose any post processing.' section, the 'Run Custom Post Processing Stylesheet' radio button is selected. Below it, the 'URL' field is populated with 'store:///strip-security-header.xsl'. Other options like 'Send SAML Logout Message (SAML 2.0 only)', 'Include a WS-Security Kerberos AP-REQ token', 'Process WS-Trust SCT STS Request', 'Add WS-Security UsernameToken', 'Generate an LTPA Token', 'Generate a Kerberos SPNEGO token', 'Request TFIM Token Mapping', 'Generate an ICRX token for z/OS Identity Propagation', 'Generate a SAML Assertion with only Authentication Statement', and 'Generate a SAML Assertion/Response' are all currently turned off. At the bottom, there are 'Back', 'Commit', and 'Cancel' buttons.

Figure 8-28 Choose post-processing

Strip security header: The incoming requests contain a UsernameToken in the WS-Security header. In certain cases, if the WS-Security header has the `mustUnderstand` attribute set to true, the Web service being invoked might not be able to process the WS-Security header so that this token is removed before forwarding the request to the back-end service. DataPower provides a pre-designed stylesheet for this situation.

11. Click **Commit**.

Content-based routing

In this scenario, we want to use a less powerful infrastructure for order requests where the total amount of the order is under a threshold, as illustrated in Figure 8-29 on page 373. The back-end MQ queue is selected based on the total order amount, which is derived from the message content by calculating the unit price times the number of orders.

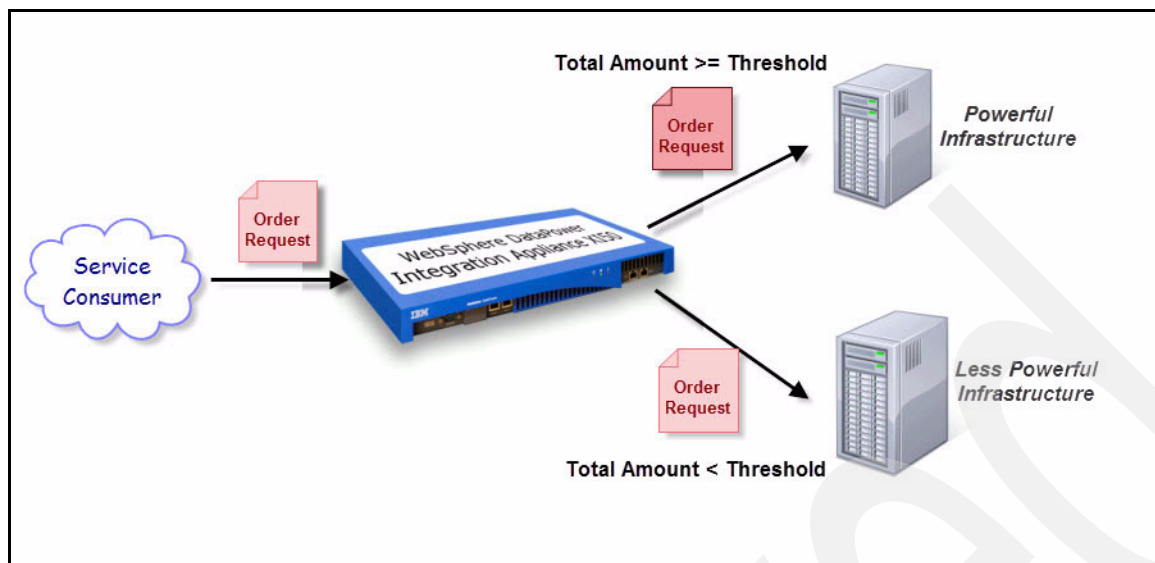


Figure 8-29 Content-based routing

There are several ways to accomplish content-based routing in DataPower:

- ▶ Matching rules can be used to select a specific processing policy, and within that policy, a back-end server is hard-coded in a routing action.
- ▶ From within a processing policy, an XPath Routing Table can be created, which uses a series of XPath expressions to determine the outbound URL.
- ▶ An XSL stylesheet can be used to generate the outbound URL.

Dynamic back-end configuration

A WS-Proxy can be either a static-back end or a dynamic-back end type:

- ▶ A *static-back end* indicates support for a single remote server. If the type is static from WSDL, the back-end configuration is from the configuration source. This configuration creates a static back-end URL by rewriting the wsdl:service address with the Remote Endpoint Rewrite Policy during the configuration and at refresh time.
- ▶ A *dynamic-back end* indicates support for multiple remote servers. The address of the target server is programmatically extracted from the client request by using extension elements. These servers are configured with the routing action in the processing policy. This setting utilizes either a DataPower variable, a stylesheet, or an XPath expression to determine the back-end URL with which the XI50 communicates. Therefore, a single processing policy can utilize multiple protocols and destinations that are dynamically decided at run time.

To perform a dynamic routing, the back-end type of the WS-Proxy must be dynamic-back end so that it informs the device that the back-end host and URL are subject to modification by the processing policy.

Perform the following steps to configure WS-Proxy as a dynamic-backend type:

1. Click the **Proxy Settings** tab, as shown in Figure 8-30 on page 374.



Figure 8-30 Configure Web Service Proxy settings

2. From the Type list, select **Dynamic Backend**, as shown in Figure 8-31.

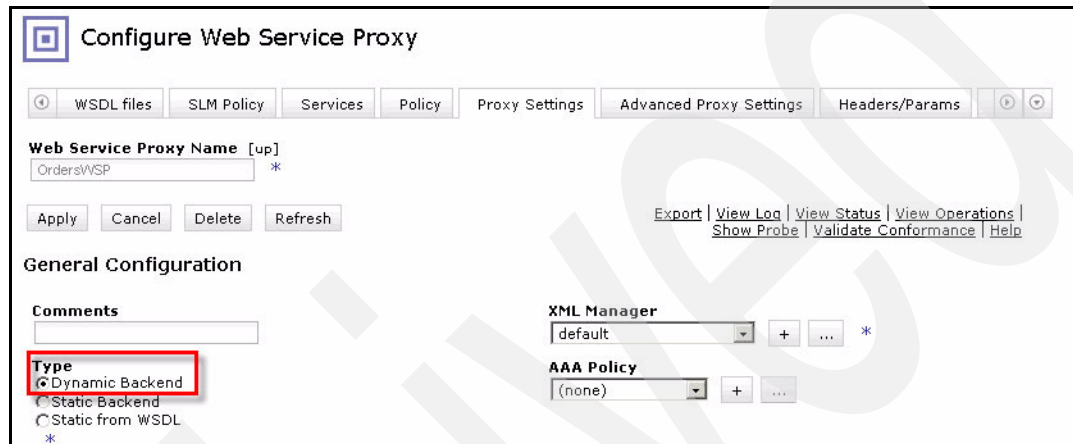


Figure 8-31 Defining the dynamic back-end behavior of WS-Proxy

Configuring the Route action

The *Route action* is used to select the desired back-end URL. There are three ways to select the back-end URL:

- ▶ **Use Stylesheet to Select Destination:** when this option is selected, the output from a designated stylesheet will be the destination URL. This option allows you to create an XSL stylesheet that includes the logic to determine and create the destination URL.
- ▶ **Use XPath to Select Destination:** When this option is selected, a table, which is referred to as an *XPath Routing Map*, of XPath/URL mappings is used to determine the back-end URL. For each row in the table, the XPath is evaluated; if it returns a non-empty nodeset, the associated URL is selected as the destination.
- ▶ **Use Variable to Select Destination:** This option allows you to specify a URL, which can be either the destination URL or the name of a DataPower variable that contains the URL.

For this scenario, we use a stylesheet for content-based routing. To configure a Route Action, perform the following steps:

1. Open the processing rule that was defined for the order creation operation by clicking **Processing Rules** on the WSDL Policy Tree Representation page.

2. Drag a **Route** action icon and drop it after the **AAA** action, as shown in Figure 8-32.

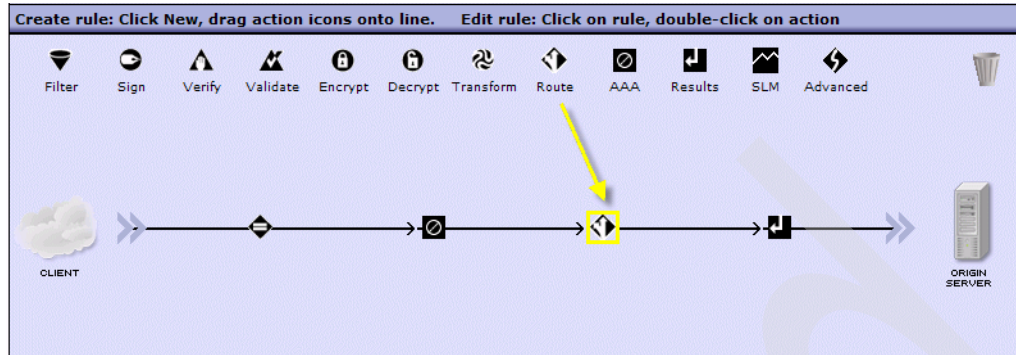


Figure 8-32 Adding the Route action

3. Double-click the **Route** action icon to display the Configure Route (Using Stylesheet or XPath Expression) Action window.
4. For the Selection Method, select **Use Stylesheet to Select Destination**. Select the stylesheet to use, as shown in Figure 8-33.

Figure 8-33 Configuring the routing selection method

Example 8-1 shows the stylesheet that was used.

Example 8-1 The stylesheet used

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:dp="http://www.datapower.com/extensions" extension-element-prefixes="dp"
exclude-result-prefixes="dp" version="1.0">
  <xsl:template match="/">
```

```

    <!-- select possible backend MQ Oueues -->
    <xsl:variable name="routeTable">
      <Services>
        <Service name="good">
<url>dpmq://ITSO.QM.ORDER/?RequestQueue=ITSO.QUEUE.ORDER.CREATE;ReplyQueue=ITSO.QUE
EUE.ORDER.CREATE.REPLY</url>
          </Service>
        <Service name="bad">
<url>dpmq://ITSO.QM.ORDER2/?RequestQueue=ITSO.QUEUE.ORDER.CREATE;ReplyQueue=ITSO.Q
UEUE.ORDER.CREATE.REPLY</url>
          </Service>
        </Services>
      </xsl:variable>

<xsl:variable name="numberItem"
select="/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and
local-name()='Envelope']/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelo
pe/' and
local-name()='Body']/*[namespace-uri()='http://www.ITS0Enterprise.com/ITS0Ordering
/' and local-name()='createOrder']/createOrder/lineItem/numberItem"/>
<xsl:variable name="unitPrice"
select="/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and
local-name()='Envelope']/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelo
pe/' and
local-name()='Body']/*[namespace-uri()='http://www.ITS0Enterprise.com/ITS0Ordering
/' and local-name()='createOrder']/createOrder/lineItem/unitPrice"/>

      <xsl:choose>
        <xsl:when test="$numberItem*$unitPrice > 1000">
          <xsl:variable name="destURL"
select="string($routeTable/Services/Service[@name='good']/url/text())"/>
          <!-- --> <xsl:message dp:priority="debug">The routing-url: <xsl:value-of
select="dp:variable('var://service/routing-url')"/></xsl:message> <!-- -->
          <dp:set-variable name="'var://service/routing-url'" value="$destURL"/>
        </xsl:when>

        <xsl:otherwise>
          <xsl:variable name="destURL"
select="string($routeTable/Services/Service[@name='bad']/url/text())"/>
          <!-- --> <xsl:message dp:priority="debug">The routing-url:
<xsl:value-of select="dp:variable('var://service/routing-url')"/></xsl:message>
          <!-- -->
          <dp:set-variable name="'var://service/routing-url'"
value="$destURL"/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:template>
  </xsl:stylesheet>

```

5. Click **Done** to save the route action configuration.
6. Click **Apply** to save the entire configuration for the WS-Proxy.

Increasing the priority level of the Ordering service

Prioritization can place certain requests ahead of others. Because creating orders requires a higher-level quality than querying orders, we assign a higher priority to the Ordering operation.

To assign a higher priority, perform the following steps:

1. Open the processing rule that was defined for WS-Proxy by clicking **Processing Rules** in the WSDL Policy Tree Representation page.
2. Click **Priority** of the ITSOOrdering Service, and for Operation Priority, select **High**, as shown in Figure 8-34.

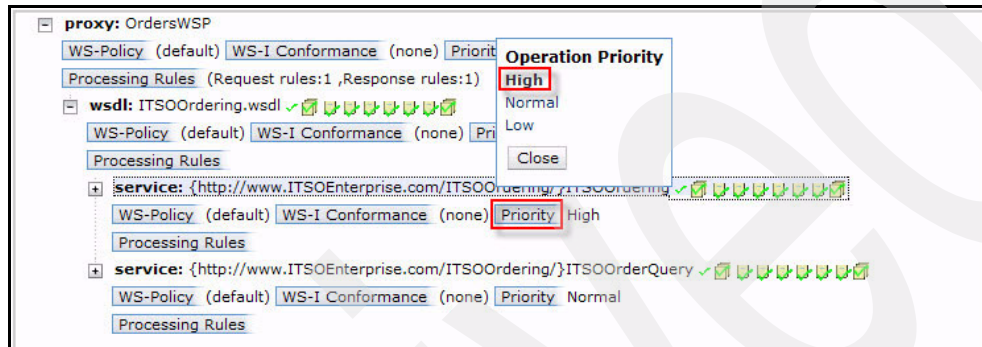


Figure 8-34 Assigning high priority

3. Click **Apply** to save the changes.

Creating the queue manager object for MQ Connection

A DataPower queue manager object acts as a reusable proxy between the DataPower appliance and an MQ Queue Manager, which resides on a remote WebSphere MQ server. The queue manager object is responsible for coordinating all communications between the DataPower appliance and WebSphere MQ. Follow these steps:

1. Log in to the DataPower WebGUI.

2. On the Control Panel, click **Objects** → **Network Settings** → **MQ Queue Manager**, as shown in Figure 8-35.

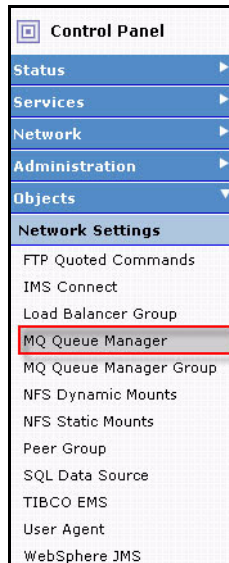


Figure 8-35 Opening the MQ Queue Manager list page

3. On the Configure MQ Queue Manager page, click **Add** to create a new MQ Queue Manager object.

4. Configure the MQ queue manager, as shown in Figure 8-36:
 - a. Set Name to **ITSO.QM.ORDER**.
 - b. Set Host Name to the host name or the IP address where your queue manager is running, followed by a port number in parentheses. Your queue manager is listening on this port. We use 9.12.5.179(1515).
 - c. Set the Queue Manager Name to your queue manager's name. We use the queue manager ITSO.QM.ORDER.
 - d. The Channel Name is prefilled with the system default channel name of SYSTEM.DEF.SVRCONN. Optionally, you can create a separate server connection channel and use it. We use SYSTEM.ADMIN.SVRCONN.

The screenshot shows a configuration window titled "Configure MQ Queue Manager". It has two tabs: "Main" and "Advanced", with "Advanced" currently selected. The window displays the following configuration details:

- MQ Queue Manager:** ITSO.QM.ORDER [up]
- Buttons:** Apply, Cancel, Delete, Undo
- Administrative State:** enabled disabled
- Comments:** SYSTEM.DEF.SVRCONN
- Host Name:** 9.12.5.179(1515) *
- Queue Manager Name:** ITSO.QM.ORDER
- Channel Name:** SYSTEM.ADMIN.SVRCONN

Figure 8-36 Configure MQ Queue Manager

5. At the top of the form in Figure 8-36, click **Apply** to create the queue manager connection object.

8.4.2 Payment Web Service Proxy

In this scenario, we configure a WS-Proxy for the payment process to off-load the security gateway functionalities from WebSphere ESB. In this scenario, we create a Web Service Proxy Service that intercepts the Web service calls to an external web service, which is provided by the Payment Company. The proxy has the following characteristics:

- ▶ The HTTPS protocol is used to secure the communication between DataPower and the external web service.
- ▶ The credit card number within the request SOAP message is encrypted.
- ▶ The request rate is monitored and a notification action is taken when the request rate reaches a certain threshold.

Figure 8-37 on page 380 shows the end-to-end processing sequence for the inbound and outbound traffic that includes the requesting payment process, DataPower, and external web service.

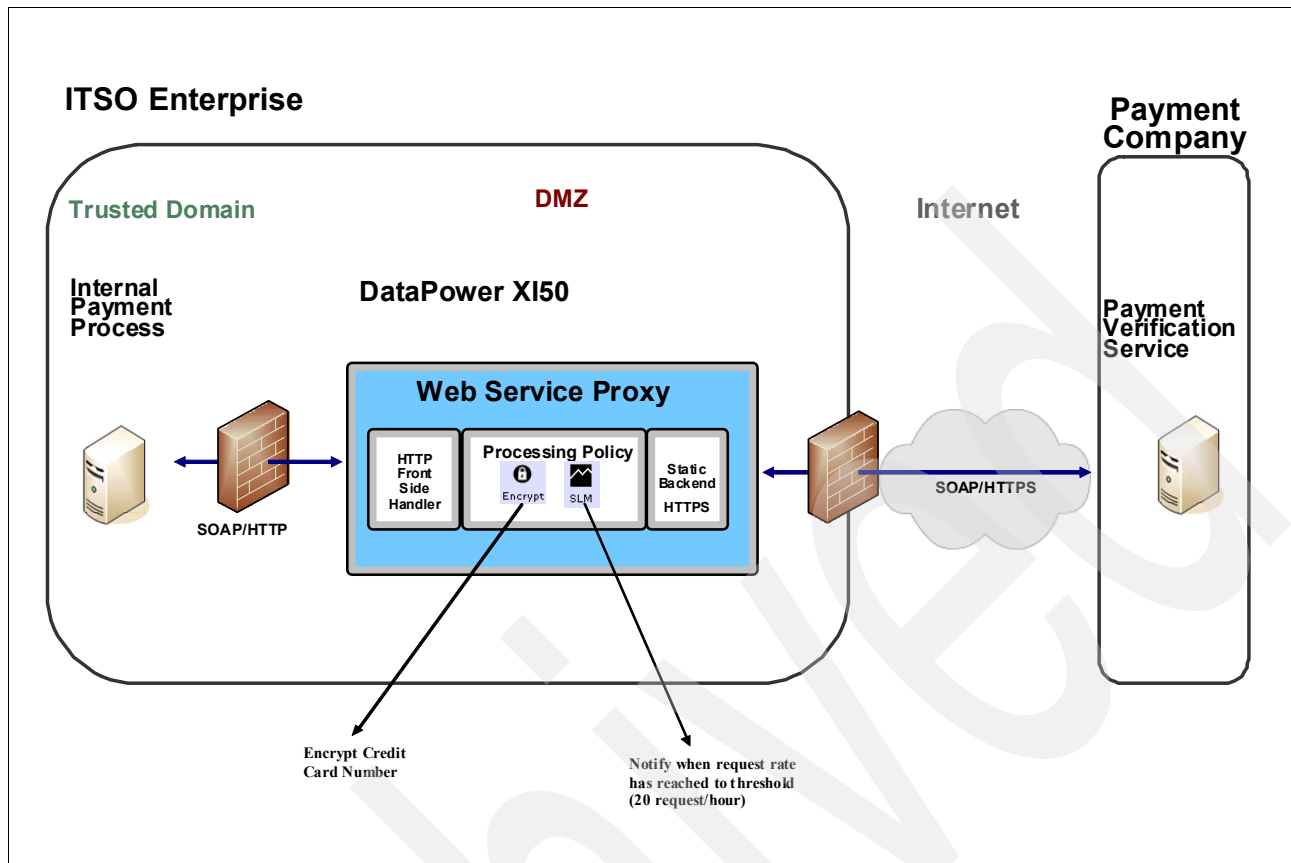


Figure 8-37 End-to-end processing for Payment Proxy

In the following sections, we only describe how to configure field-level encryption and service-level monitoring. Refer to “Creating the Web Service Proxy” on page 356 for the configuration of WS-Proxy.

Field-level encryption

Message encryption is used to provide privacy; intercepted messages cannot be read easily, if at all. By encrypting the message content, the privacy of the content becomes decoupled from the transport mechanism. For example, messages that are sent over an SSL connection are encrypted and, thus, have a certain degree of privacy, but after the message exits the SSL connection, no further privacy is provided. By encrypting the content of the message, the message can travel across transport boundaries, such as between HTTP and MQ, for example, and remain private.

The DataPower device can encrypt all or part of the message content, in accordance with the relevant specifications. This function might be useful when a message passes through intermediate handlers. The intermediaries can read only those parts of the message that relate to them but not the whole message. This approach can help with routing, for example.

A message is encrypted with the recipient’s certificate, so that only the recipient can decrypt the message using the corresponding private key, which only the recipient possesses.

To configure field-level encryption, perform the following steps:

1. Open the processing rule that is defined for WS-Proxy by clicking **Processing Rules** in the WSDL Policy Tree Representation page, as shown in Figure 8-38.

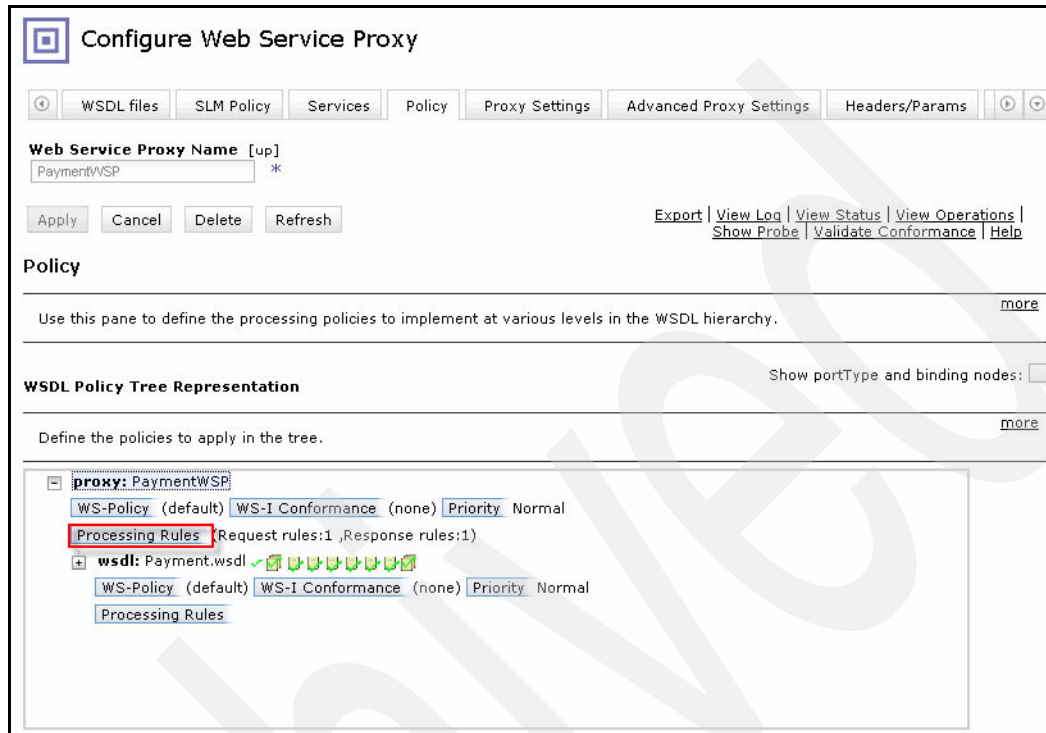


Figure 8-38 Opening Processing Rules

2. Drag and drop the **Encrypt** action icon, as shown in Figure 8-39.

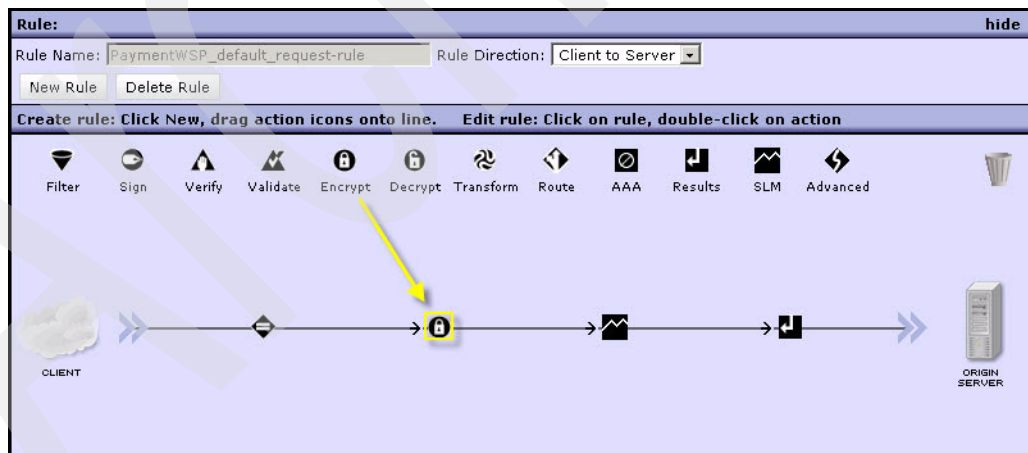


Figure 8-39 Drag and drop the Encrypt action

3. Double-click the **Encrypt** action icon to display the Configure Encrypt Action window.

- For Message Type, choose **Selected Elements (Field-Level)**. Notice that after you make this choice, a new field, Document Crypto Map, appears, as shown in Figure 8-40.

Figure 8-40 Configuring the Encrypt Action Message Type as field level

Document crypto map: The *document crypto map* is used to tell the encryption action how to find the elements to encrypt. Because the document is in XML, the most natural way of selecting the target elements is with an XPath expression. The document crypto map represents a collection of XPath expressions that are executed to find the target elements to encrypt. Because we want to encrypt the `<cardNumber>` element, we create a document crypto map that contains this XPath expression.

- Click the plus sign (+) next to the Document Crypto Map drop-down menu. The Configure Document Crypto Map page appears.
- Type a Name for the document crypto map.
- Click **Xpath Tool** next to the XPath Expression field for guidance to build the XPath expression, as shown in Figure 8-41. The “Build XPath Expression from sample XML File” page appears.

Figure 8-41 Opening XPath Tool for guidance to build the XPath expression

8. For URL of Sample XML Document, browse for the sample XML file, **PaymentRequest.xml**. The page refreshes with the Content of sample XML file section filled with the selected sample XML file.
9. Click the **cardNumber** element of the sample XML File. The Xpath expression is automatically generated in the Selected XPath Expression box, as shown in Figure 8-42.

Build XPath Expression from sample XML File [Help](#)

Select sample XML document

URL of Sample XML Document: local:/// PaymentRequest.xml *

Namespace Handling: local prefix uri

Selected XPath Expression

XPath *

```
//*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and local-name()='Envelope']/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and local-name()='Body']/*[namespace-uri()='http://PaymentCompany/Payment' and local-name()='requestPayment']/cardNumber
```

Content of sample XML file.
Click on an element, attribute name, or attribute value to select an XPath expression.

```
<?xml version='1.0' encoding='UTF-8'>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:pay="http://PaymentCompany/Payment">
<soapenv:Header />
<soapenv:Body>
<pay:requestPayment>
<cardNumber>1234567</cardNumber>
<totalAmount>200000</totalAmount>
</pay:requestPayment>
</soapenv:Body>
</soapenv:Envelope>
```

Figure 8-42 Building the XPath expression

10. Click **Done** to save the XPath expression.
11. Click **Apply** to save the document crypto map.

12. For Recipient Certificate, browse for **orderskeys**, as shown in Figure 8-43.

The screenshot shows the 'Encrypt' configuration dialog box with the following settings:

- Envelope Method:** WSSec Encryption (selected), Standard XML Encryption, Advanced.
- Message Type:** SOAP Message, Raw XML Document, Selected Elements (Field-Level) (selected), Advanced.
- Document Crypto Map:** OrdersCryptoMap (selected), +, ... *
- Asynchronous:** on (selected), off.
- Message and Attachment Handling:** Message Only (selected), Save.
- Encryption Key Type:** Use Ephemeral Key Transported by Asymmetric Algorithm (selected), Save.
- Recipient Certificate:** orderskey (selected and highlighted with a red box), +, ..., Save.
- WS-Security Version:** 1.0 (selected), Save.
- Use Dynamically Configured Recipient Certificate:** on (selected), off, Save.
- One Ephemeral Key:** on (selected), off, Save.

Figure 8-43 Selecting the recipient certificate

13. Click **Done** to save the changes to the Encrypt action.

14. Click **Apply** to save the changes to the WS-Proxy.

Configuring the service-level monitor

ITSO Enterprise IT Department services often require a specific level of quality in terms of response time, throughput, and availability. The IT infrastructure has to provide a means to measure the current status and react appropriately if the specified thresholds of the indicators are reached, thus ensuring the QoS that is expected by the customers.

The service-level monitoring (SLM) capabilities in DataPower offer the following benefits:

- ▶ Avoiding abuse and overuse of services and secure services from denial-of-service (DoS) attacks
- ▶ Enforcing service-level agreements (SLAs)
- ▶ Differentiating between users with varying needs for service quality
- ▶ Defining actions to be taken if thresholds for indicators are reached, for example, sending notifications if there is bad response time or too many error messages from the back end
- ▶ Optimizing response times and sharing resources adequately among the parties that access the services

In this scenario, we monitor the request rate of the payment verification service, because ITSO Enterprise pays a certain amount of money for every request. We configure an SLM request rule, which counts all user transactions during the specified interval. After reaching the defined limit during an interval, the SLM request rule enforces the defined action.

To configure a SLM request rule, follow these steps:

1. On the Configure Web Service Proxy page, click the **SLM Policy** tab, as shown in Figure 8-44.



Figure 8-44 SLM Policy tab

2. Navigate to the appropriate level of the WSDL tree and click **Request**, as shown in Figure 8-45.

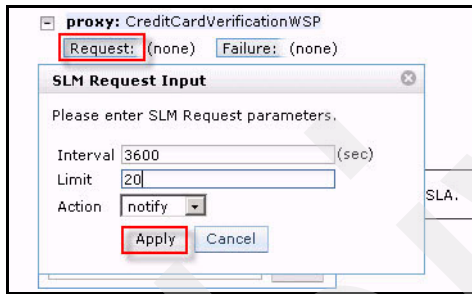


Figure 8-45 SLM Request rule

3. In the SLP Request Input box, define the request rule (Figure 8-45).
4. Click **Apply**.

8.4.3 Creating a multi-protocol gateway

A single multi-protocol gateway can support the definitions of multiple front-side handlers. Data can be input to the DataPower XI50 using separate protocols. To implement Scenario 2, we create a multi-protocol gateway with two front-side handlers: MQ and HTTP.

To create a new Multi-Protocol Gateway, follow these steps:

1. Log in to the ITSOEnterprise domain.
2. Click the **Control Panel** link to display the Control Panel.

3. Click the **Multi-Protocol Gateway** icon, which is located on the top row of service icons, as shown in Figure 8-46.

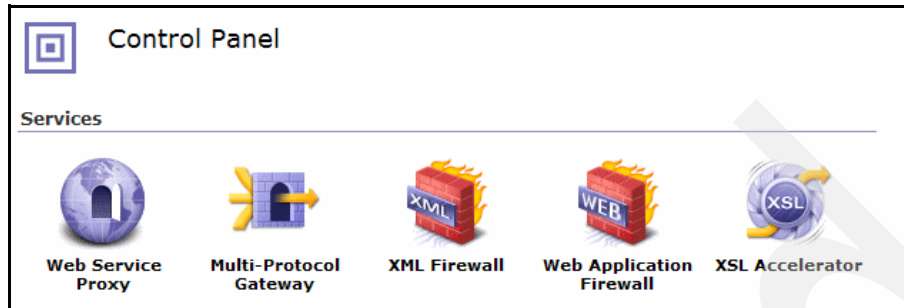


Figure 8-46 Control Panel

4. On the General Configuration page, configure the general settings as described:
 - a. In the Multi-Protocol Gateway Name field, type `OrdersMPG`.
 - b. For Type, select **dynamic-backend**. In this type, the dynamic back end, back-end server, and port are determined by the stylesheet in a policy action.

After applying the changes, the General Configuration section is shown in Figure 8-47.

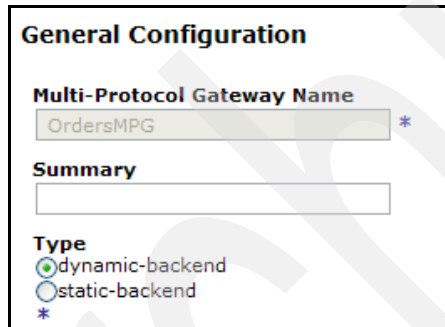


Figure 8-47 General Configuration section

5. Next, we configure the front-side handler settings by creating two front-side handlers: the MQ Handler and the HTTP Handler. To create the MQ Front Side Handler, perform the following steps:
- On the Front side settings window, click the plus sign (+) and select **MQ Front Side Handler** from the list. The Configure MQ Front Side Handler window opens, as shown in Figure 8-48.

Configure MQ Front Side Handler
This configuration has been added and not yet saved.

Main

MQ Front Side Handler

Apply Cancel

Name *

General

Administrative State enabled disabled

Comments

Queue Manager (none) + ... *

Get Queue *

Put Queue

The number of concurrent MQ connections 1

Figure 8-48 Configure MQ Front Side Handler window

- On the Configure MQ Front Side Handler window, enter the following information:
 - For Name, type MQFSH_ITS0_JMS.90045.
 - For Get Queue, type ITS0.JMS.90045.QUEUE.
- In the Queue Manager field, click the plus sign (+) and select **MQ Queue Manager**, as shown in Figure 8-49 on page 388.

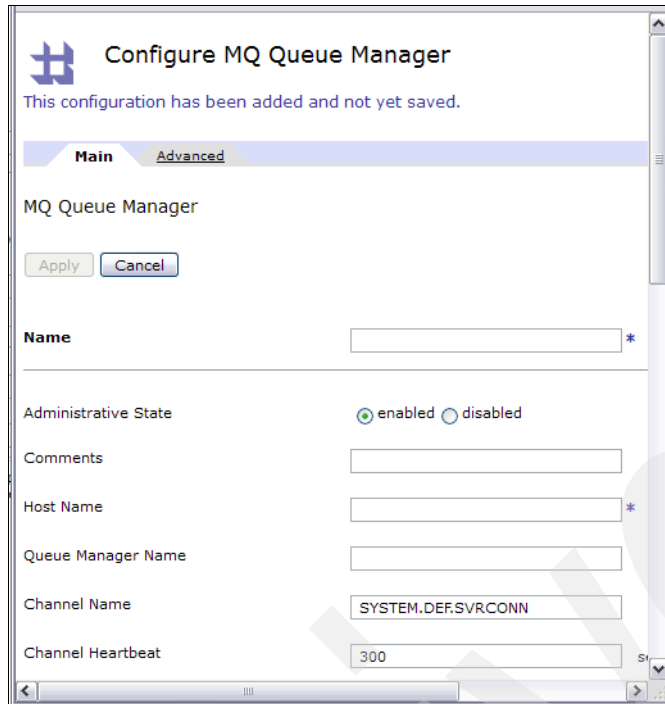


Figure 8-49 Configure MQ Queue Manager

- d. In the Configure MQ Queue Manager window (Figure 8-49), enter the following information:
 - For Name, type `WAREHOUSE.AGENT.QM`.
 - For Host Name, type `9.5.12.213(1414)`.
 - e. Click **Apply**.
 - f. Select **WAREHOUSE.AGENT.QM** from the drop-down box.
 - g. Click **Apply**.
6. Add the newly created MQ Front Side Handler **MQFSH_ITSO_JMS.90045** to the list of Front Side Handlers by clicking **Add** on the Front side settings window.

7. Next, we create the HTTP Front Side Handler:
 - a. On the Front side settings window, click the plus sign (+) and select **HTTPfront side handler** from the list that is shown. The window that is shown in Figure 8-50 opens.

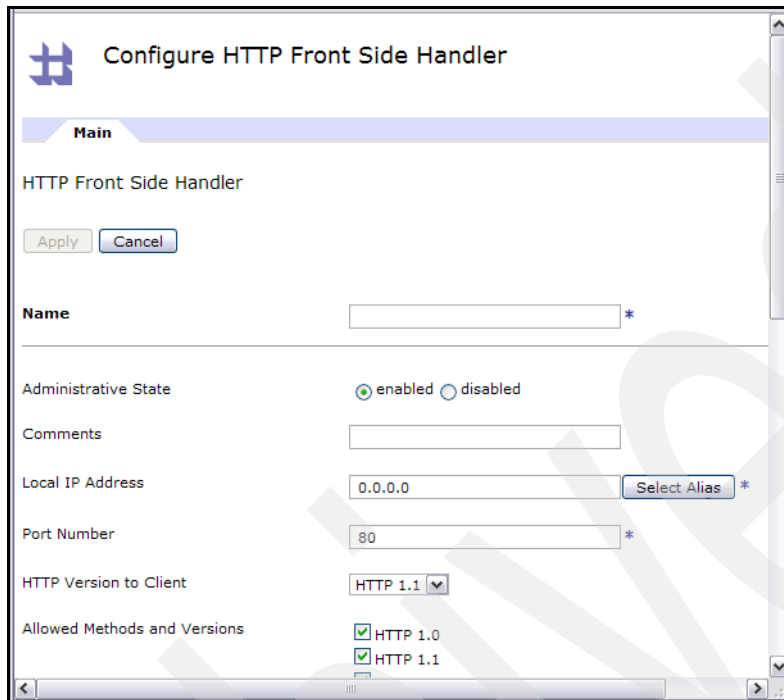


Figure 8-50 Configure HTTP Front Side Handler

- b. In the Configure HTTP Front Side Handler window (Figure 8-50), enter the following information:
 - For Name, type `SupplierHTTP`.
 - For Port Number, type `1234`.
- c. Click **Apply**.
- d. Add the newly created HTTP Front Side Handler **SupplierHTTP** to the list of front-side handlers by clicking **Add** of the Front side settings window.

At this point, we have created two front-side handlers and they appear in the Front side settings window, as shown in Figure 8-51.

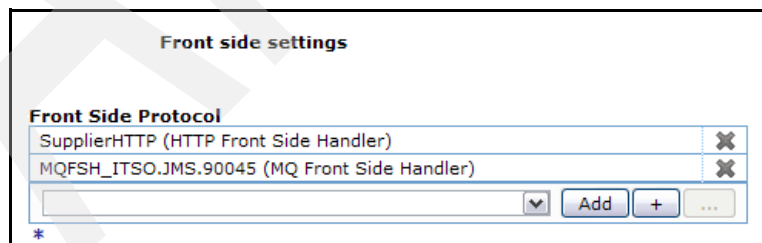


Figure 8-51 Front side settings window

8. We need to create the Gateway Processing Policy. Perform the following steps to create this policy:

- a. On the Configure Multi-Protocol Gateway window, click the plus (+) sign next to Multi-Protocol Gateway Policy.
- b. For the policy name, type OrdersTestMPG_Policy1.

After creating the Gateway Processing Policy, we need to create three rules for this policy: one rule for handling Web service requests, a second rule for polling messages from a queue, and a third rule for “match all”, which simply echoes the message.

9. Create SupplierWSRule. SupplierWSRule encrypts the messages and routes them to a supplier-provided endpoint, which is, in this case, is `http://9.12.5.225:8088/SOAPNodesSampleMessageSetSOAP_HTTP_Service`. Use the following steps:

- a. Create a new rule by clicking **New Rule**. The window in Figure 8-52 opens.
- b. For the Rule Name, type the rule SupplierWSRule.
- c. For Rule Direction, select **Client to Server**.
- d. Create the processing actions **Encrypt**, **Route**, and **Results**. Complete and save the rule, as shown in Figure 8-52 and Figure 8-53.

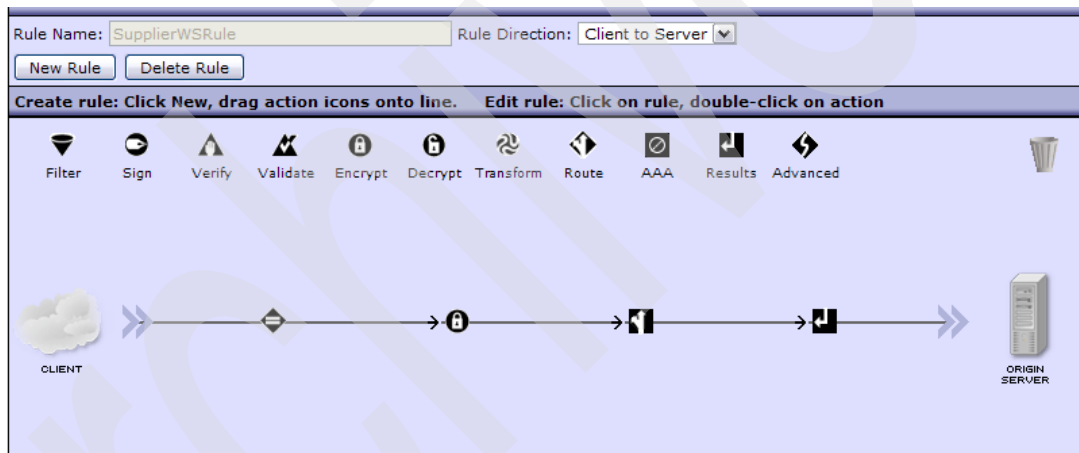


Figure 8-52 Create SupplierWSRule

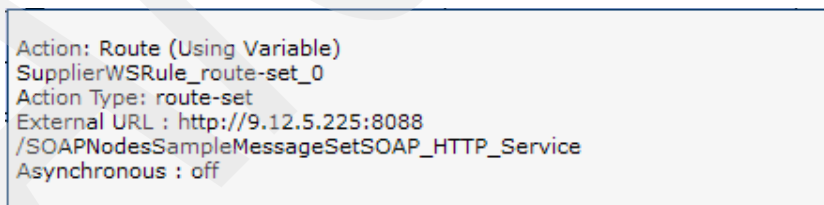


Figure 8-53 Create the processing actions

10. Create MQRule. MQRule polls messages from the MQ queue ITSO.JMS.90045.QUEUE, encrypts them, and routes them to the supplier-provided MQ endpoint, which, in this case, is ITSO.JMS.90045.SUPPLIER.QUEUE. Perform the following steps:
 - a. Create a new rule by clicking **New Rule**.
 - b. For Rule Name, type MQRule.
 - c. For Rule Direction, click **Client to Server**.
 - d. Create the processing actions **Encrypt**, **Route**, and **Results**. Complete and save the rule, as shown in Figure 8-54 and Figure 8-55.

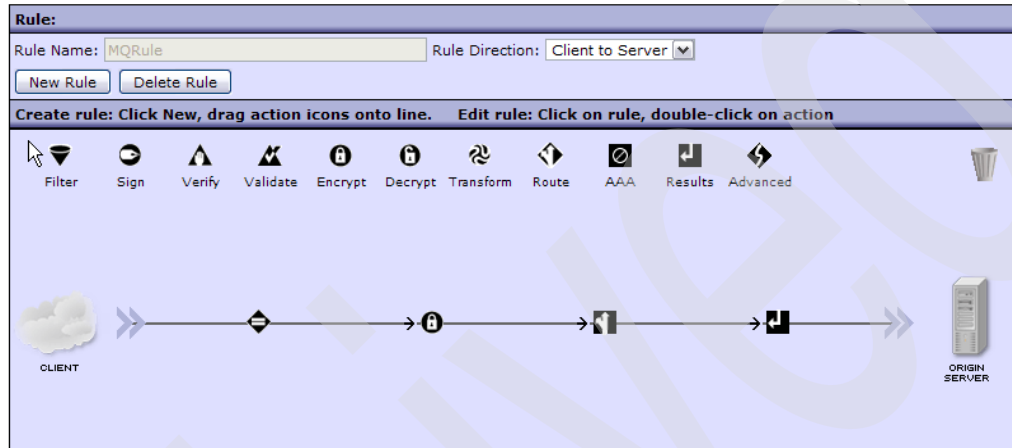


Figure 8-54 Create MQRule

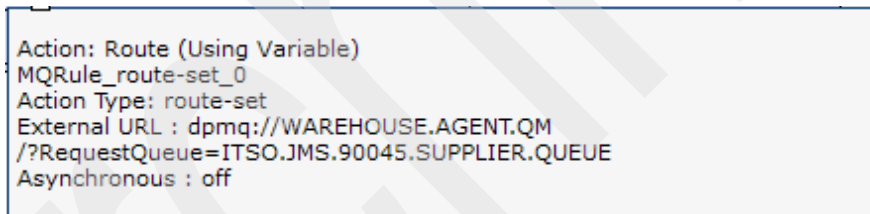


Figure 8-55 Create the processing actions

11. Create OrdersTestMPG_Policy1_rule_0 using the following steps:
 - a. Create a new rule by clicking **New Rule**.
 - b. For the Rule Name, type OrdersTestMPG_Policy1_rule_0.
 - c. For the Rule Direction, select **Both Directions**.
 - d. Create the processing actions **Match** and **Results**. Complete and save the rule, as shown in Figure 8-56.

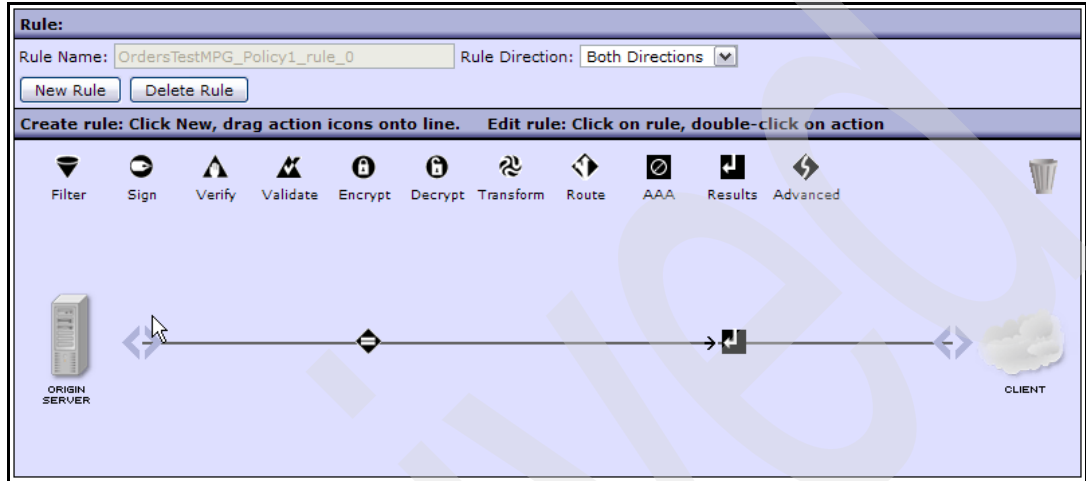


Figure 8-56 Create OrdersTestMPG_Policy1_rule_0

After all three rules have been created, the gateway policy OrdersTestMPG_Policy1 shows three rules, as shown in Figure 8-57.

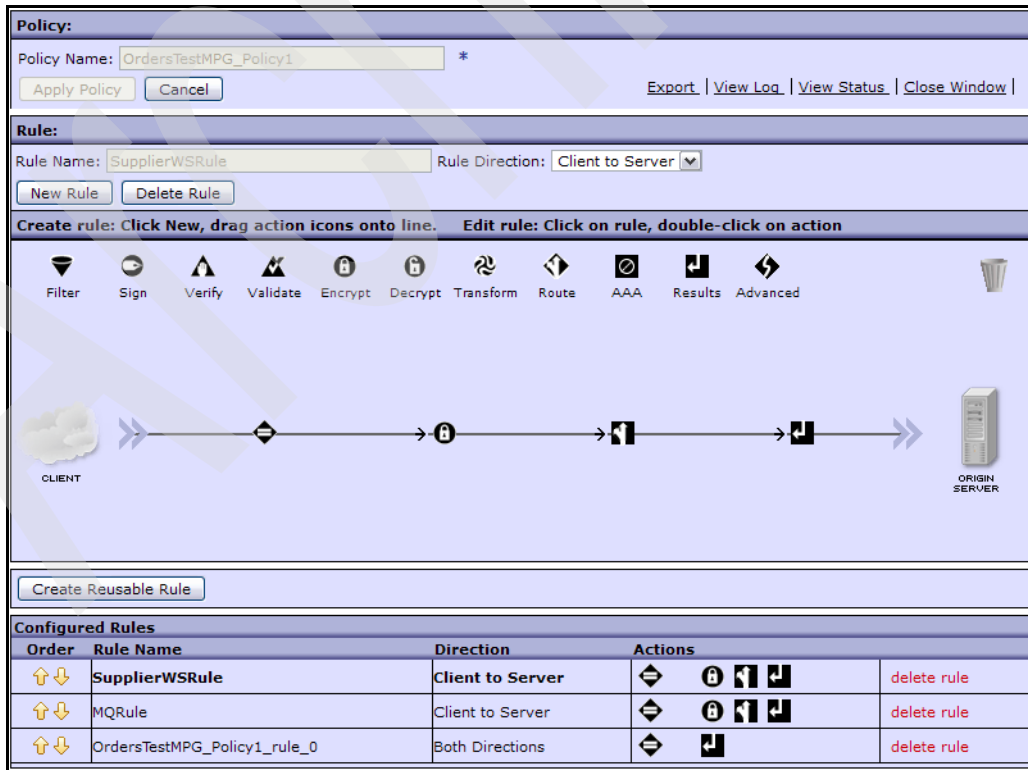


Figure 8-57 Three rules shown

12. Configure the User agent settings. On the User Agent settings window, click the SSL Crypto Profile drop-down list box and select **PaymentClientCryptoProfile**, as shown in Figure 8-58.

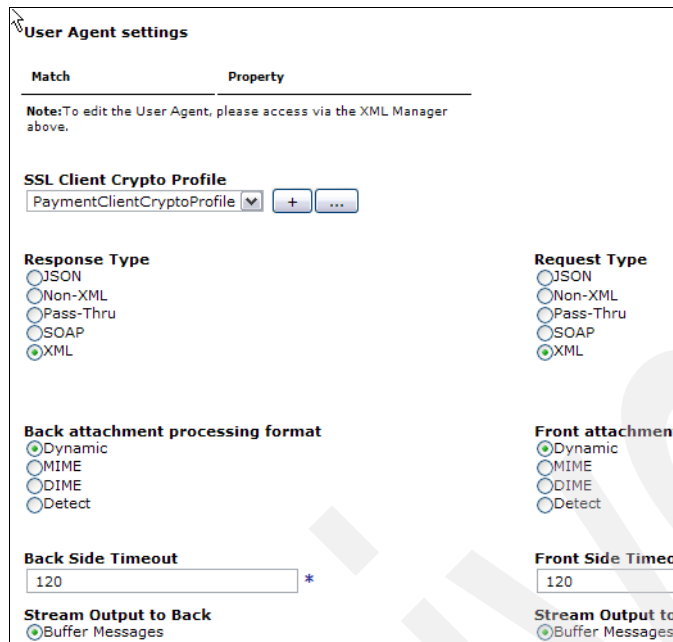


Figure 8-58 Select PaymentClientCryptoProfile

13. Click **Apply** in the upper left of the Configure Multi-Protocol Gateway window to create and save the OrdersMPG service, as shown in Figure 8-59.

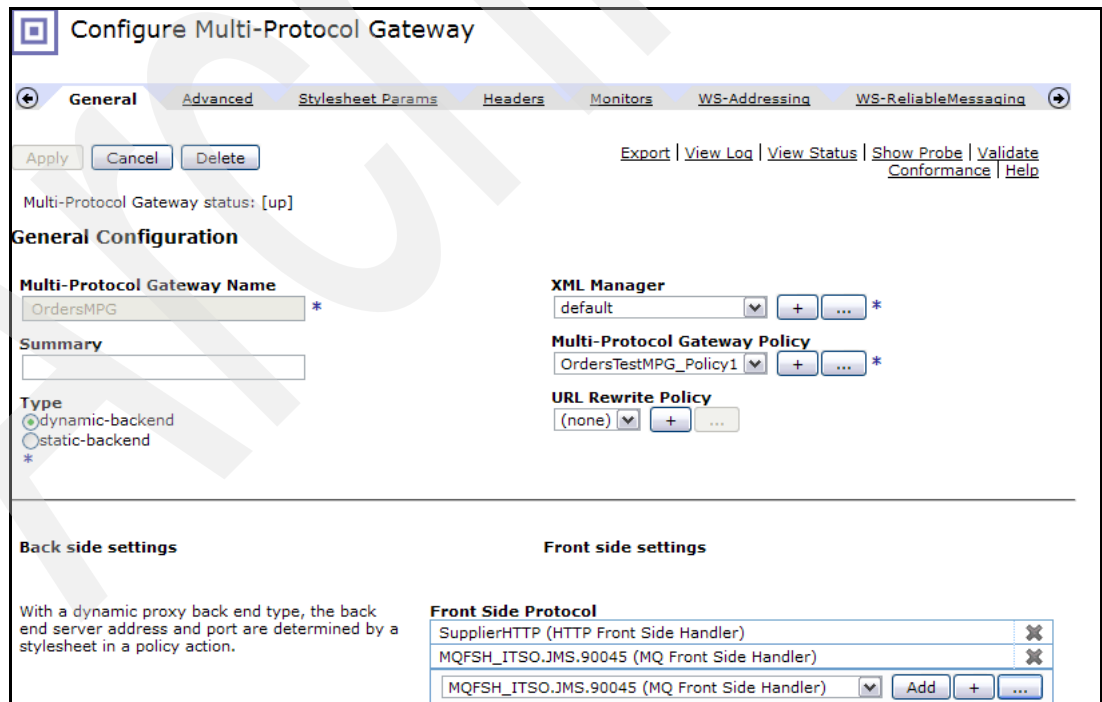
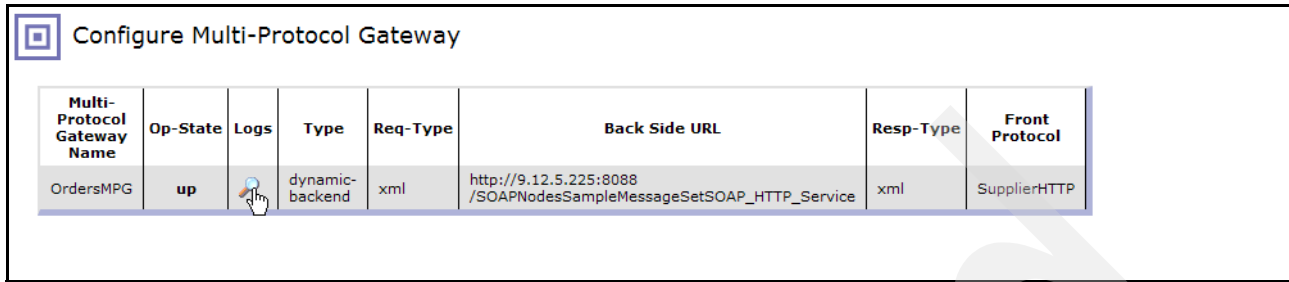


Figure 8-59 Configure Multi-Protocol Gateway window

After the service is fully configured, the OrdersMPG service is listed on the Configure Multi-Protocol Gateway window, which is shown in Figure 8-60.



The screenshot shows a window titled "Configure Multi-Protocol Gateway" with a table containing one row of data. The table has the following columns: Multi-Protocol Gateway Name, Op-State, Logs, Type, Req-Type, Back Side URL, Resp-Type, and Front Protocol.


Multi-Protocol Gateway Name	Op-State	Logs	Type	Req-Type	Back Side URL	Resp-Type	Front Protocol
OrdersMPG	up		dynamic-backend	xml	http://9.12.5.225:8088/SoapNodesSampleMessageSetSOAP_HTTP_Service	xml	SupplierHTTP

Figure 8-60 Configure Multi-Protocol Gateway window

8.5 Summary

This chapter described two scenarios: the Order Service is exposed to external partners and customers by implementing a Web Service Proxy and securing order fulfillment and status messages is achieved by implementing a Multi-Protocol Gateway.

Conclusion

In this book, you have traced the service-oriented architecture (SOA) adoption path of the ITSO Enterprise company. You have seen how this company went from a siloed approach to data access and to an environment of governed accessibility and reusable business services. ITSO Enterprise freed itself from a batch dependency and can now execute critical business functions in real time. The services that the company has built position the company well, for maximum return on investment (ROI) and reduction in implementation times as ITSO Enterprise adds more and more projects to the SOA. This is Smart SOA.

We discuss the following topics in this chapter:

- ▶ 9.1, “Benefits of Smart SOA” on page 396
- ▶ 9.2, “Business achievements” on page 396
- ▶ 9.3, “What is next for ITSO Enterprise in the Smart SOA ladder” on page 398
- ▶ 9.4, “Resources” on page 398

9.1 Benefits of Smart SOA

Smart SOA does not make a company wait until an ideal architectural end state is achieved to derive benefits. Instead, Smart SOA maintains a focus on business needs and pain points throughout its path to maturity. This approach allows enterprises to take incremental steps through the adoption of proven patterns, and gain immediate value at each stage while progressing toward the most flexible and agile architecture possible.

Smart SOA techniques provide increased responsiveness and flexibility to handle new customer demands, market conditions, competitive threats, government regulation compliance, and acquisitions. It places a greater emphasis on reuse to reduce and contain the cost of creating business solutions rapidly. In addition to the reuse value of enterprise-wide services, it also emphasizes the SOA business value of increased flexibility and agility.

9.2 Business achievements

We have illustrated two distinct paths to SOA adoption at this one company. First, a programmatic path of new development allows ITSO Enterprise to build a foundation for additional business optimization capabilities, such as business process management (BPM), complex event processing, and business rules management. Second, an organic path evolved its existing investments and technical capabilities into a service-oriented environment with maximum efficiency.

Each step in these paths offers unique and tangible business benefits, from the straightforward cost and quality improvements of Managed File Transfer, to the innovative customer relationship enhancements that are provided by real-time situational awareness. Smart SOA's intelligence is realized by its ability to achieve significant business results that are enabled by state of the art technology.

9.2.1 Achievement from scenario 1

In Chapter 2, "Service Enablement pattern" on page 23, ITSO Enterprise achieved a single view of customers' data in real time. Multiple data sources were updated through a single aggregated service. The solution was implemented through the Smart SOA Service Enablement pattern. This category of pattern addresses the problem of encapsulating functionality and presenting it through a service-oriented interface. It enables the implementation of an enterprise service bus (ESB) that delivers access to services derived from functionality implemented using diverse technologies across the enterprise and to present a single, consistent view of these services to the service consumers. This approach creates a number of business benefits, including improved customer serviceability, better data quality presented to the customer, and ultimately reduced operating costs in the enterprise.

9.2.2 Achievement from scenario 2

In Chapter 3, "Service Enablement pattern: Enterprise Content Management System" on page 75, ITSO Enterprise achieved service enablement of its enterprise contents. This capability allows ITSO Enterprise to access the enterprise content through exposed services. The solution was implemented through the Smart SOA Service Enablement pattern. In this scenario, ITSO Enterprise went a step further by building a complex service facade using richer features, such as adapter integrations in ESB. This pattern is often used to build

higher-level business or enterprise services from lower level IT services, components, or fine-grained application function. Through the service enablement of access to digitized documents, ITSO Enterprise has laid the foundation for moving costly manual document handling processes to more efficient, cost-effective automated processes that can be accessed through many channels.

9.2.3 Achievement from scenario 3

In Chapter 4, “Service Reuse and Governance pattern” on page 149, ITSO Enterprise was able to leverage its previous investments in adding new business capability to its IT ecosystem. Existing services were used by new users and the access to these services was governed by using a runtime governance model that was implemented through WebSphere Services Registry and Repository. This capability helped ITSO Enterprise reuse an existing application functionality within an enterprise, hence reducing the overall application maintenance costs, and making the functionality available to a wide variety of users.

9.2.4 Achievement from scenario 4

In Chapter 5, “Message Privacy Enforcement pattern” on page 201, ITSO Enterprise ensured the integrity and confidentiality of its sensitive data while in transit between applications in the environment. This capability helped ITSO Enterprise to comply with its industry’s security standard (similar to Payment Card Industry Data Security Standard (PCI DSS)). It was done without code changes to its applications, and introduced minimal overhead to its transaction processing times.

9.2.5 Achievement from scenario 5

In Chapter 6, “Enterprise Messaging and File Backbone pattern” on page 233, ITSO Enterprise introduced Managed File Transfer capabilities to its environment. By consolidating its file movement and providing auditing and integrity, ITSO Enterprise has reduced its overall integration costs, and it has been able to improve the quality of service for the dependent business activities that rely on these files. ITSO Enterprise has also laid the groundwork to start migrating scheduled, latent batch processing to real-time business services.

9.2.6 Achievement from scenario 6

In Chapter 7, “Evolve to SOA pattern” on page 317, ITSO Enterprise took a step further and implemented an end-to-end inventory management and supplier ordering automation system. This system was implemented using Smart SOA file processing and protocol conversion patterns. WebSphere Message Broker was used as an ESB to implement the listed patterns.

By connecting its file-based processing with its real-time integration layer, ITSO Enterprise has further enabled the reuse of existing assets in its infrastructure and taken the benefits of Smart SOA to the world of file-based processing.

9.2.7 Achievement from scenario 7

In Chapter 8, “Hybrid Bus pattern” on page 353, ITSO Enterprise focused on providing protection from data tampering. ITSO Enterprise focused on authentication, authorization, encryption, and non-repudiation functionality to the existing ESB services. This step allowed ITSO Enterprise to extend the reach of its existing services to more highly secure internal channels, and in the future, this capability can be the groundwork for opening up partner

channels for further service reuse and the potential for new revenue sources through a cloud-based delivery of services.

9.3 What is next for ITSO Enterprise in the Smart SOA ladder

ITSO Enterprise took an organic approach to maximize the efficiency of its existing infrastructure. Now, ITSO Enterprise has established an excellent Smart SOA foundation, which will help it to grow and adapt seamlessly in the future.

Using the Smart SOA patterns, ITSO Enterprise has achieved the following benefits:

- ▶ Reduced spending on redundant development and support through the implementation and reuse of services and other IT functions
- ▶ Increased efficiencies through the deployment of the correct reusable services
- ▶ Enhanced clarity of business ownership responsibilities and governance
- ▶ Reduced manual tasks and administration by introducing automation

After fulfilling the foundational requirements, ITSO Enterprise has to look beyond and identify ways to dynamically implement and manage the business processes of its organization. This approach will help ITSO Enterprise to survive in the ever-changing competitive market, where the company has to closely watch and govern its business processes.

Business process management (BPM) combines business processes, information, and IT resources, aligning an organization's core assets—people, information, technology, and processes—to create a single integrated view, with real-time intelligence, of both its business measurements and IT system performance.

BPM will enable ITSO Enterprise to be flexible and responsive to ever-changing on-demand business through the optimization and automation of the business processes to perform these tasks:

- ▶ Identify and eliminate redundancies and bottlenecks
- ▶ Decouple business integration logic from its underlying implementation code
- ▶ Increase portability and decrease maintenance cost by being based on industry standards
- ▶ Automate process implementation and eliminate manual deployment tasks
- ▶ Execute new business rules and processes immediately
- ▶ Visualize actual process performance against key performance indicators
- ▶ Pinpoint future process improvements

To get more details about IBM offerings in the BPM area, refer to this website:

<http://www-01.ibm.com/software/info/bpm/>

Also, you can refer to the industry-accepted BPM pattern, which is discussed in *Patterns: SOA Foundation - Business Process Management Scenario*, SG24-7234.

9.4 Resources

There is no doubt that the landscape of enterprise application integration is an increasingly complex space, and without good guidance, solutions can be daunting. The approaches outlined in this book are intended to help give you that map to SOA success. These patterns

and methods are based on the experience of IBM and the thousands of clients with which we have partnered to realize that success. For more information and to connect with our experts, visit the resources that are listed in Table 9-1.

Table 9-1 IBM resources for Smart SOA

Topic	References
IBM SOA	http://www-01.ibm.com/software/solutions/soa/index.html
WebSphere Application Integration	http://www-01.ibm.com/software/websphere/products/appintegration/
developerWorks SOA	http://www.ibm.com/developerworks/webservices/
Global WebSphere community	http://www.websphereusergroup.org/

Archived

Additional material

This book refers to additional material that can be downloaded from the Internet as described in the following sections.

Locating the web material

The web material associated with this book is available in softcopy on the Internet from the IBM Redbooks web server. Point your web browser at:

<ftp://www.redbooks.ibm.com/redbooks/SG247944>

Alternatively, you can go to the IBM Redbooks website at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG247944.

Using the web material

The additional web material that accompanies this book includes the following files:

<i>File name</i>	<i>Description</i>
SG247944_Ch4.zip	Zipped configuration files and programs used in Chapter 4
SG247944_Ch5.zip	Zipped configuration files and programs used in Chapter 5
SG247944_Ch6.zip	Zipped configuration files and programs used in Chapter 6
SG247944_Ch7.zip	Zipped configuration files and programs used in Chapter 7

System requirements for downloading the web material

The Web material requires the following system configuration:

Hard disk space: 10 MB minimum
Operating System: Windows/Linux

Downloading and extracting the web material

Create a subdirectory (folder) on your workstation, and extract the contents of the web material .zip file into this folder.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks publications

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- ▶ *Patterns: SOA Foundation - Business Process Management Scenario*, SG24-7234
- ▶ *Getting Started with IBM WebSphere Process Server and IBM WebSphere Enterprise Service Bus Part 1: Development*, SG24-7608
- ▶ *Getting Started with IBM WebSphere Process Server and IBM WebSphere Enterprise Service Bus Part 2: Scenario*, SG24-7642
- ▶ *Getting Started with IBM WebSphere Process Server and IBM WebSphere Enterprise Service Bus Part 3: Run time*, SG24-7643

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- ▶ IBM Smart SOA website:
<http://www.ibm.com/software/solutions/soa/>
- ▶ IBM WebSphere Service Registry and Repository features:
http://www-01.ibm.com/software/integration/wsrr/features/index.html?S_CMP=wspace
- ▶ IBM WebSphere Application Server Network Deployment website:
<http://www.ibm.com/software/webservers/appserv/was/network/>
- ▶ IBM WebSphere MQ website:
<http://www-01.ibm.com/software/integration/wmq/>
- ▶ IBM WebSphere MQ File Transfer Edition website:
<http://www-01.ibm.com/software/integration/wmq/filetransfer/>
- ▶ IBM DB2 website:
<http://www-01.ibm.com/software/data/db2/>

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



Redbooks

Smart SOA Connectivity Patterns: Unleash the power of IBM WebSphere Connectivity Portfolio

(0.5" spine)
0.475" x 0.873"
250 <-> 459 pages



Smart SOA Connectivity Patterns

Unleash the Power of IBM WebSphere Connectivity Portfolio



Redbooks®

Learn how to unleash the power of WebSphere Connectivity Portfolio

Experiment with real-life scenarios within the business context

Understand the Smart SOA connectivity patterns

This IBM Redbooks publication provides you with a path to demystify the complexity of adopting a service-oriented architecture (SOA) approach to integrating applications and services. With an iterative evolution of a fictitious company, which is called ITSO Enterprise, we demonstrate several scenarios about how we can implement an IBM Smart SOA approach that helps ITSO Enterprise to achieve its business goals to be a global interconnected enterprise, one step at a time.

It is not our intention to dive into the extremely technical details of every product or to tell you specific solutions for specific problems, but rather, to advise you about how to look at these problems from a business context perspective and then to provide you with a concise deployment using the IBM WebSphere Connectivity portfolio of products to easily address them.

This book will be a reference for IT Specialists and IT Architects working on implementing Smart SOA solutions using the IBM WebSphere Connectivity portfolio of products at client sites, as well as for decision makers, IBM employees, IBM Business Partners, and IT Managers.

**INTERNATIONAL
TECHNICAL
SUPPORT
ORGANIZATION**

**BUILDING TECHNICAL
INFORMATION BASED ON
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:
ibm.com/redbooks**

SG24-7944-00

ISBN 0738435988