MetaSolv Solution ™ 6.0

XML API Integration Developer's Reference

Eleventh Edition

August 2008

METASOLV®
SOFTWARE

Copyright and Trademark Information

Document History

Edition	Date	Reason
First	March 2005	FCS
Second	March 2005	(FCS) Incorrect procedure sequence for workflow
Third	June 2005	Listing of methods, parameters, and return values
Fourth	October 2005	Addition of LSR XML API and the Performance chapter.
Fifth	December 2005	Addition of methods for CNAM and LIBD
Sixth	April 2006	Addition of SOA XML API and new methods for existing XML APIs
Seventh	September 2006	Addition of new XML API methods. Also, the format of the appendix that contains the XML API method information has been changed.
Eighth	December 2006	Addition of new Appendix: Navigating the XSD. Removed sections that were covered by Setup Guide.
Ninth	March 2007	Added explaination of the three levels of xsds.
Tenth	June 2007	Updated Copyrights, About this Guide chapter, and customer portal references with Oracle. Added information on MSLVwliPool to Chapter 1. Added new control methods under Inventory Management and LSR10 to Chapter 2 and Appendix B. Added information regarding special characters to Chapter 1.
Eleventh	August 2008	Appended new workflow Billing Telephone Number in Appendix D and XML API Annotated Schemes in 02_IntegrationOverview chapter.

Contents

About this guide v
Audience v
Additional information and help v
Oracle Support vi
MetaSolv Solution documentation set vi
Additional documentation resources viii

1. Setting up 1
Technical requirements and installation instructions 1
About the development database 1
Recommended deployment configurations 2
JMS messaging requirements 4
.... 5

2. Integration Overview 7
About the MetaSolv Integration and Portal Toolkit 8
Controls 8
WebLogic controls 9
MetaSolv Solution controls 10
MetaSolv Solution schema 14
WebLogic Platform 8.1 17
Basic integration steps 18
Special characters 19

3. Developing an integration application 21
Planning the application 24
Creating a new application in Workshop 24
Adding the MetaSolv Solution controls to Workshop 27
Creating data transformations 28
Request transformation control 28
Response transformation control 33
Building the workflow 34
Step 1: Creating the workflow process file 34
Step 2. Adding controls to the Workshop Data Palette 36
Step 3. Specifying how the request is invoked 38
Step 4. Adding a group to the workflow 40
Step 5. Adding the request transformation method 41
Step 6. Adding the method to process the request 44
Step 7. Adding the response transformation method 46
Step 8. Setting up exception handling 50
Testing the application in Workshop 61
Creating a build 63

4. Post Development Tasks 65
Updating the production database 65

Creating the SQL script 65
Running the SQL script 66
Setting up gateway events 67
Creating a gateway event 67
Configuring the gateway.ini file 68

5. Troubleshooting 71

6. Performance 77

Setting up to precompile workflows 77

Appendix A: XML API sample code 81

Where to find the sample files 81
Setting up the sample code 82
Upgrading sample files 84
Viewing the samples in Workshop 86
Composite sample 88

Appendix B: Control interfaces 89

Appendix C: Navigating the XSD 105

Example 1: importCustomerAccount 107
Example 2: getCustomerAccountByKey 114
Example 3: createEntityByValueRequest 121

Appendix D: XML API methods 127

Customer Management API 128
importCustomerAccount 128
getCustomerAccountByKey 129
deleteCustomerRequest 130
Order Management API 131
queryOrderManagementRequest 132
startOrderByKeyRequest 133
updateOrderManagementRequest 134
getOrderByKeyRequest 135
createOrderByValueRequest 135
assignProvisionPlanProcedureRequest 136
getActivationDataByKeyRequest 136
transferTaskRequest 137
updateE911DataRequest 137
getE911DataRequest 138
updateEstimationCompletedDateRequest 138
addTaskJeopardyRequest 139
getTaskDetailRequest 139
TaskJeopardyRequest 140
getPSROrderByTN 140
processSuppOrder 141
getCNAMDataRequest 142
getLidbDataRequest 142

updateCNAMDataRequest 143
updateLidbDataRequest 143
reopenTaskRequest 144
createAttachmentRequest 144
createOrderRelationshipRequest 145
processBillingTelephoneNumber 146
Inventory Management API 147
createEntityByValueRequest 147
getServiceRequestDLRsValue 148
getEntityByKeyRequest 148
updateEntityByValueRequest 149
queryInventoryManagementRequest 150
updateTNRequest 150
tnRecall 151
tnValidationRequest 151
auditTrailRecording 152
getNetworkAreasByGeoAreaRequest 152
getNetworkComponentsRequest 153
getIpAddressesRequest 153
createInventoryAssociationRequest 154
createNewInventoryItemRequest 154
queryNetworkLocation 155
queryEndUserLocation 155
getLocationRequest 156
deleteLocationRequest 156
updateLocationRequest 157
createLocationRequest 157
getAvailablePhysicalPortsRequest 158
Service Order Activation API 159
createSOAMessageRequest 159
getSoaTnsForOrderRequest 159
getSoaDefaultsRequest 160
getSoaInformationRequest 160
getSoaMessageToSendRequest 161
setTnSoaCompleteRequest 161

About this guide

This guide explains how to use the MetaSolv Integration and Portal Toolkit to integrate MetaSolv Solution with other MetaSolv Products and with external applications. The toolkit provides a workspace and other tools for the integration development and testing.

Audience

This guide is for individuals who are responsible for using the MetaSolv Integration and Portal Toolkit in a development environment and developing software to integrate an external application or another MetaSolv product with MetaSolv Solution. This guide assumes the reader has a working knowledge of Oracle 9i, Windows XP Professional, BEA WebLogic Platform 8.1, and Java J2EE.

Additional information and help

To get additional information or help for MetaSolv Solution, refer to the following resources:

- Oracle E-Delivery—Provides access to product software and documentation.
  - Visit the E-Delivery Web site at http://edelivery.oracle.com.
  - Software and product documentation are contained in the Oracle Communications MetaSolv Solution 6.0 Media Pack.
  - Developer documentation is contained in the Oracle Communications MetaSolv Solution Developer Documentation Pack. Access to developer documentation requires a password.
- Oracle MetaLink—Provides access to software patches and a searchable Knowledge Base.
  - Visit the MetaLink Web site at https://metalink.oracle.com/, and log on using your User Name and Password.
  - Click the Patches & Updates tab to search for patches (efixes).
  - Click the Knowledge tab to search for technical bulletins, fixed issues, and additional product information. To narrow your search, click the Communication Apps link under Product Categories on the left side of the page.

Oracle Support

The preferred method of reporting service requests (SRs) is through MetaLink. MetaLink is available 24 hours a day, 7 days a week.

Although it is Oracle's preference that you use MetaLink to log SRs electronically, you can also contact Support by telephone. If you choose to contact Support by phone, a support engineer will gather all the information regarding your technical issue into a new SR. After the SR is assigned to a technical engineer, that person will contact you.

For urgent, Severity 1 technical issues, you can either use MetaLink or you can call Support. Oracle Support can be reached locally in each country. To find the contact information for your country, go to http://www.oracle.com/support/contact.html.

MetaSolv Solution documentation set

This guide is one book in a set of documents that helps you understand and use MetaSolv Solution. Figure 1 shows the complete documentation set.

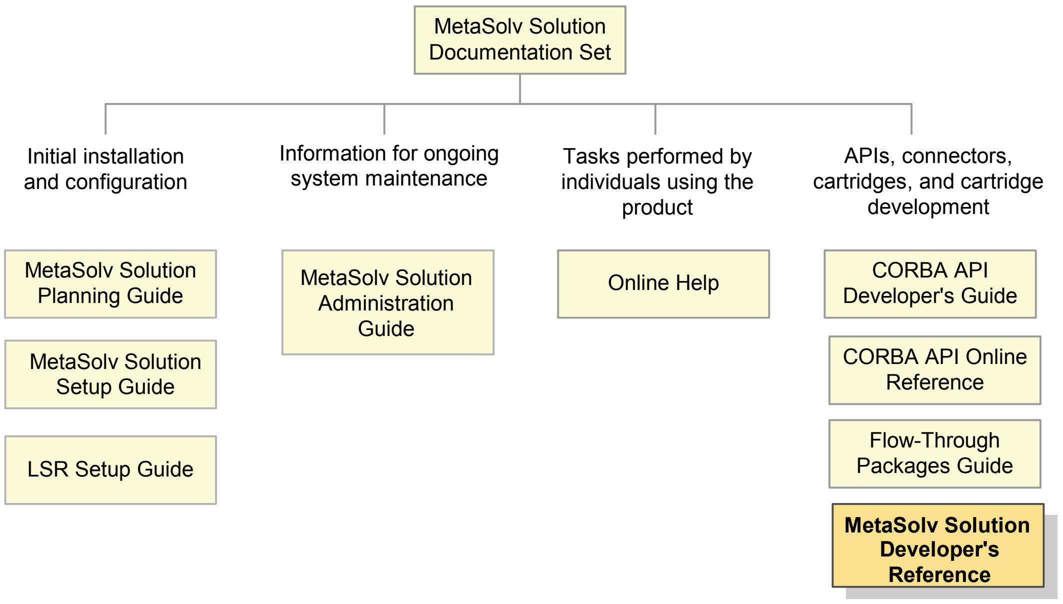

MetaSolv Solution Documentation Set
Initial installation and configuration
Information for ongoing system maintenance
Tasks performed by individuals using the product
APIs, connectors, cartridges, and cartridge development
MetaSolv Solution Planning Guide
MetaSolv Solution Setup Guide
LSR Setup Guide
MetaSolv Solution Administration Guide
Online Help
CORBA API Developer's Guide
CORBA API Online Reference
Flow-Through Packages Guide
MetaSolv Solution Developer's Reference

Figure 1: MetaSolv Solution documentation set

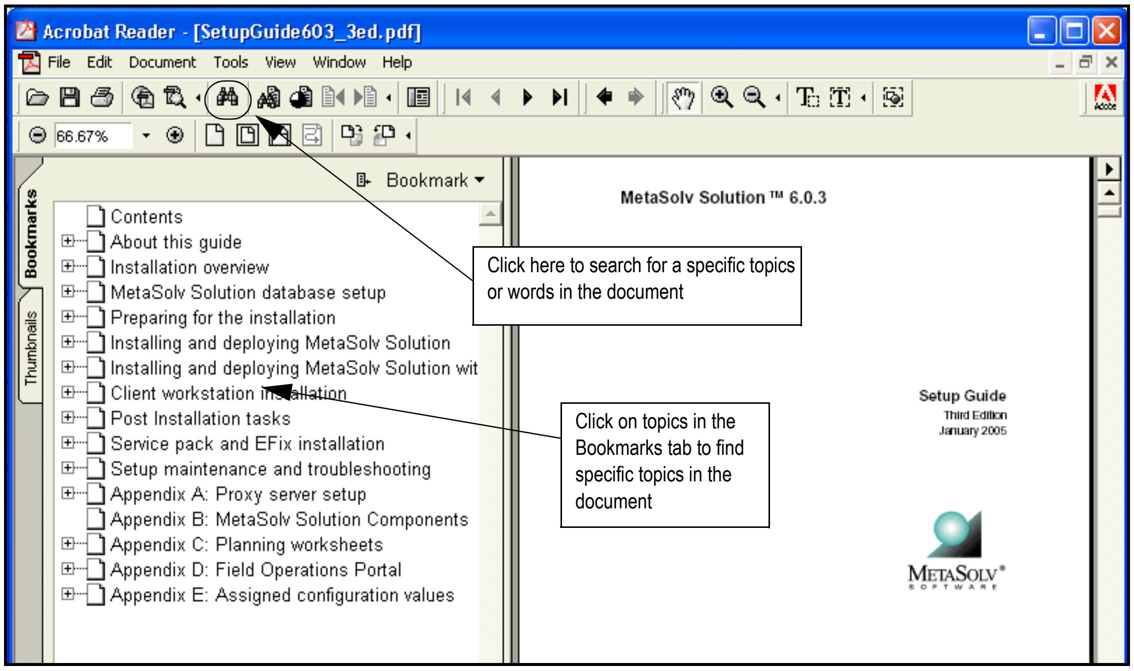
MetaSolv Solution books are delivered in Portable Document Format (PDF). You can view a book online using Adobe Acrobat Reader.

To view a document

Locate the document on the Oracle E-Delivery or Oracle MetaLink Web site and do one of the following:

- Right-click the PDF file and select **Open** from the pop-up menu.
- Double-click the PDF file.

This action starts Acrobat Reader and opens the PDF document you selected. The following figure shows how a document appears in Acrobat Reader:


Acrobat Reader - [SetupGuide603_3ed.pdf]
File Edit Document Tools View Window Help
66.67%
Bookmark
Bookmarks
Thumbnails
Contents
About this guide
Installation overview
MetaSolv Solution database setup
Preparing for the installation
Installing and deploying MetaSolv Solution
Installing and deploying MetaSolv Solution wit
Client workstation installation
Post Installation tasks
Service pack and EFix installation
Setup maintenance and troubleshooting
Appendix A: Proxy server setup
Appendix B: MetaSolv Solution Components
Appendix C: Planning worksheets
Appendix D: Field Operations Portal
Appendix E: Assigned configuration values
MetaSolv Solution ™ 6.0.3
Setup Guide
Third Edition
January 2005
METASOLV SOFTWARE

Click here to search for a specific topics or words in the document

Click on topics in the Bookmarks tab to find specific topics in the document

Figure 2: Finding information in a PDF document

Additional documentation resources

You can obtain additional information about the XML APIs used to integrate MetaSolv Solution with other applications from the following resources:

- **XML Schema**—The XML schema used in integration for MetaSolv Solution have documentation included directly in the schema.
- **Sample code**—The sample code is installed with the MetaSolv Solution installation on the workstation if you have the XML API option.

1

Setting up

This chapter contains general information on getting ready to develop an integration application. It does not contain installation instructions for MetaSolv Solution.

Technical requirements and installation instructions

- See the technical requirements for the MetaSolv Solution application and client in the *MetaSolv Solution 6.0 Planning Guide*.
- To find complete installation instructions for the MetaSolv Solution Integration and Portal Toolkit, see the chapter entitled "Installing MetaSolv Solution with the XML API option" in the *MetaSolv Solution 6.0.3 (or higher) Setup Guide XML API Option*. Regarding the installation, note the following:
  - Single server installation is required for development.

    Clustered server installation is not available for development. The installation program for MetaSolv Solution is the same for production and development. The BEA configuration is different only in the selection of Production or Development mode.
  - The connection pool MSLVwliPool must be established.

    Connection pooling is a technique used for sharing server resources among requesting clients. This allows for multiple clients to share a cached set of connection objects that provide access to a database. The MSLVwliPool is used by calls generated from the XML APIs and is mapped to the username APP_INT. This means that any records created or updated in the M6.0.x database that resulted from a XML API call will have the last_modified_userid field set to APP_INT.

About the development database

BEA allows you to accept a default PointBase database when you configure the domain. MetaSolv recommends that you use a test Oracle database. A tool is provided that allows you to create an SQL script file that can be run against your production database to re-create any new tables created in the development database. For information on the tool, see "Updating the production database" on page 65.

Recommended deployment configurations

All of the development components can be installed and run on a single machine. Components of the development environment include:

WebLogic Integration—WebLogic Server with the WebLogic Integration extensions included.

WebLogic Workshop IDE—An integrated development environment used to develop and test customer integration and Web GUI applications.

MetaSolv Solution with the XML API option—This application has a minimum deployment of the MetaSolv Solution and its client. The client is used to set up gateway events in the MetaSolv Solution application, and it requires a Windows environment.

The following figure shows a Windows development environment.

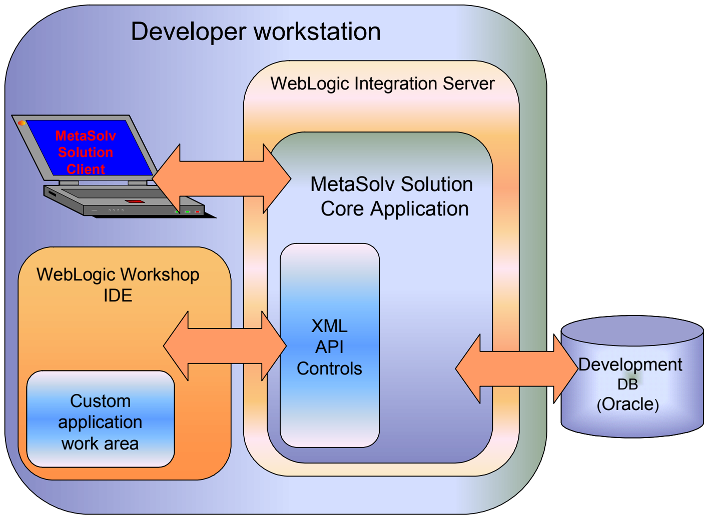


Figure 3: Windows developer workstation environment

In a Windows environment, all WebLogic components and the MetaSolv Solution core and client can be installed on the same developer workstation. The XML API controls are included in the MetaSolv Solution installation.

The following figure shows a UNIX development environment.

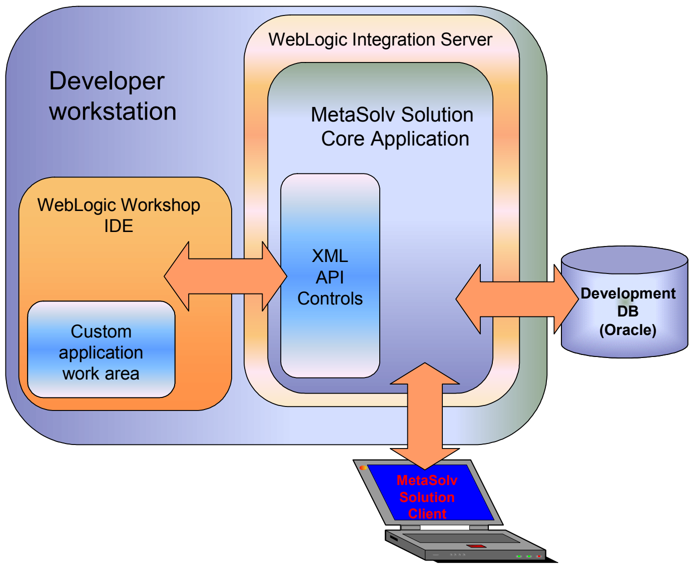


Figure 4: UNIX developer workstation environment

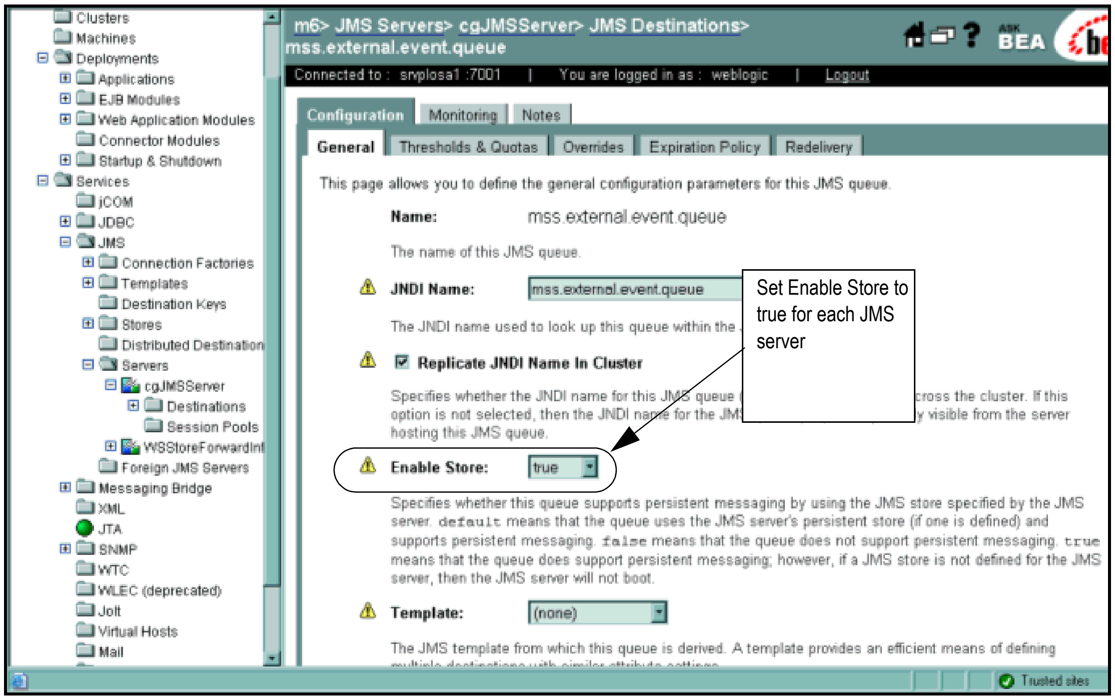
The MetaSolv Solution client requires a Windows environment so it must be loaded onto a separate machine in a UNIX development environment.

JMS messaging requirements

The MetaSolv Solution installation program sets up a paging store for each JMS server. The paging store is used exclusively for paging out non-persistent messages for the JMS server and its destinations.

The installation program also sets the **Enable Store** option to *true*. This setting is for persistent messages, which are necessary for a guaranteed message delivery system. If you create additional JMS destinations (queues/topics) after installation, you must set the **Enable Store** option to true for these destinations manually.

The following figure shows the **Enable Store** setting in WebLogic Console.



Connected to : srvplosa1 :7001 | You are logged in as : weblogic | Logout

This page allows you to define the general configuration parameters for this JMS queue.

Name: mss.external.event.queue

The name of this JMS queue.

JNDI Name: mss.external.event.queue

The JNDI name used to look up this queue within the

✓ **Replicate JNDI Name In Cluster**

Specifies whether the JNDI name for this JMS queue cross the cluster. If this option is not selected, then the JNDI name for the JM y visible from the server hosting this JMS queue.

Enable Store: true

Specifies whether this queue supports persistent messaging by using the JMS store specified by the JMS server. `default` means that the queue uses the JMS server's persistent store (if one is defined) and supports persistent messaging. `false` means that the queue does not support persistent messaging. `true` means that the queue does support persistent messaging; however, if a JMS store is not defined for the JMS server, then the JMS server will not boot.

Template: (none)

The JMS template from which this queue is derived. A template provides an efficient means of defining

Set Enable Store to true for each JMS server

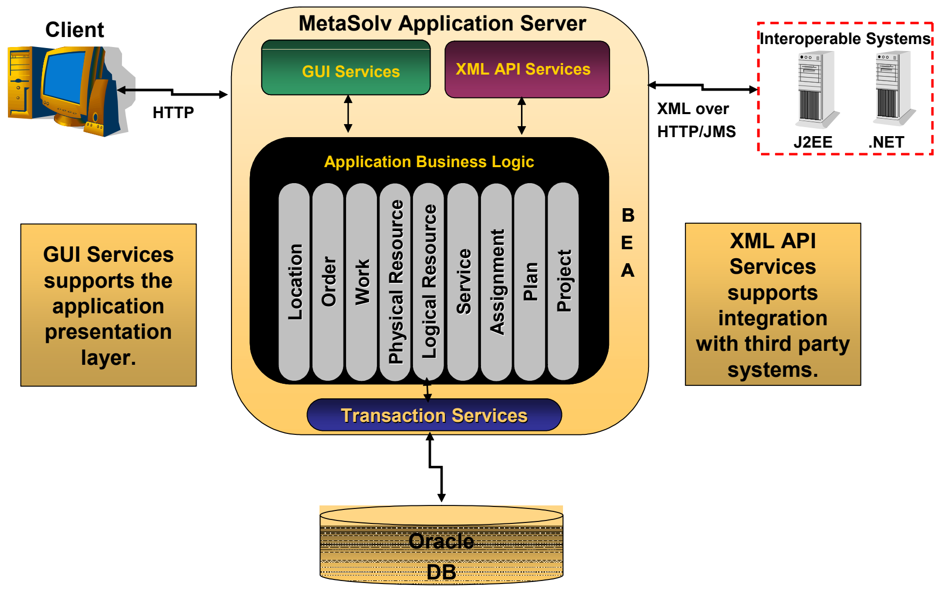
Integration Overview

This chapter provides basic information about the MetaSolv Integration and Portal Toolkit and how you can use it to integrate with MetaSolv Solution.

High Level Overview

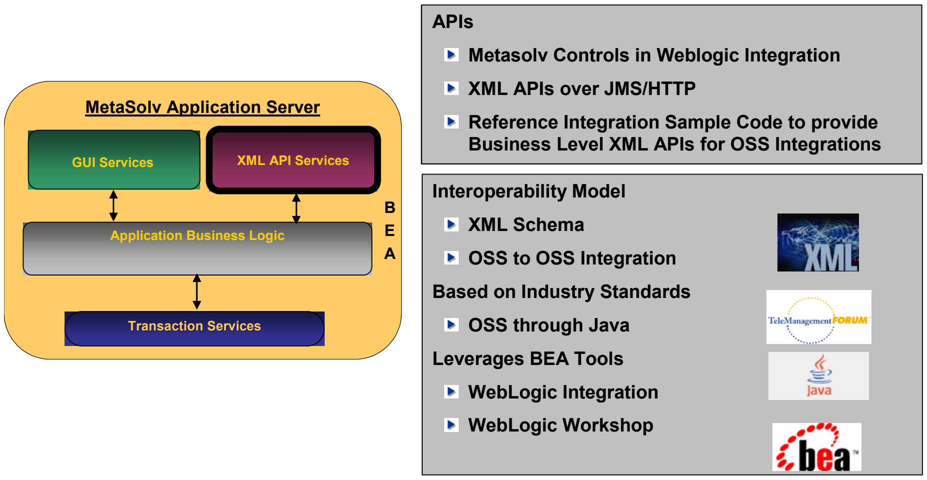
The figure below represents a high level overview of the MetaSolv Integration. At it's core is the Application Business Logic, which is grouped by functional area such as Location, Order, Work, etc. The Application Business Logic is used by both the GUI Services and the XML API Services. GUI Services supports the application presentation layer to the client over HTTP. The XML API Services supports integration with third party systems using XML over HTTP or JMS. JMS is the recommended choice because it is a more reliable messaging service. Collectively, the MetaSolv Application Server runs on a BEA WebLogic Server.

Figure 5: MetaSolv Solution Integration Overview



The figure below includes information regarding tools utilized by the MetaSolv Solution Integration, and standards that it followed.

Figure 6: MetaSolv Solution Integration Tools and Standards



APIs

- Metasolv Controls in Weblogic Integration
- XML APIs over JMS/HTTP
- Reference Integration Sample Code to provide Business Level XML APIs for OSS Integrations

Interoperability Model

- XML Schema
- OSS to OSS Integration

Based on Industry Standards

- OSS through Java

Leverages BEA Tools

- WebLogic Integration
- WebLogic Workshop

About the MetaSolv Integration and Portal Toolkit

The MetaSolv Integration and Portal Toolkit is a package that allows you to integrate a third-party software product with MetaSolv Solution using XML APIs. The toolkit includes:

- Controls
- Schema
- WebLogic Platform 8.1 SP5

  This is a third-party software product that can be purchased as an option with MetaSolv Solution. It provides an integration environment, transformation mapping, and a high-level interface that represents code elements visually in the work area in WebLogic Workshop.

The following sections describe the components of the Integration and Portal Toolkit.

Controls

A Java control is code that forms a reusable component that can be used anywhere within a platform application. The MetaSolv Solution controls referred to in this document are individual applications that expose XML APIs for integration purposes.

WebLogic Workshop, a component of WebLogic Platform, provides Java controls that you can use to encapsulate business logic and to access enterprise resources such as databases, legacy applications, and web services.

WebLogic controls

WebLogic allows the use of three different types of Java controls:

- **Built-in controls**

  Built-in controls, included in the WebLogic Workshop, provide easy access to enterprise resources. For example, the Database control makes it easy to connect to a database and perform operations on the data using simple SQL statements, while the EJB control lets you easily access an EJB. Built-in controls provide simple properties and methods for customizing their behavior, and in many cases you can add methods and callbacks to further customize a control.

  For more information on BEA built-in controls, see the BEA documentation at http://e-docs.bea.com.

- **Portal controls**

  A portal control is a type of built-in Java control designed for the portal environment. If you are building a portal, you can use portal controls to expose tracking and personalization functions in multi-page portlets.

- **Custom controls**

  You can also build a custom Java control from scratch. A custom control can act as the nerve center of a piece of functionality, implementing the desired overall behavior and delegating subtasks to built-in Java controls (and/or other custom Java controls). This use of a custom Java control ensures modularity and encapsulation. Web services, JSP pages, or other custom Java controls can simply use the custom Java control to obtain the desired functionality, and changes that may become necessary can be implemented in one software component instead of many.

  The MetaSolv Solution XML API controls were developed in WebLogic's integration environment as custom controls. The controls contain code that transforms the XML input into the proper format for MetaSolv Solution and transforms the response that returns from MetaSolv Solution into the proper format for the third-party application.

MetaSolv Solution controls

Each control works with a specific portion of the MetaSolv Solution functionality. The MetaSolv Solution controls in the Integration and Portal Toolkit include:

- Customer Management
- Order Management
- Inventory Management
- LSR Management
- Service Order Activation
- Event Management

The controls correspond to XML APIs. When you add a control into WebLogic Workshop to begin integration development, the methods under each control become available to use on the Workshop work area. You can drag the methods to the Workshop work area to create nodes in the workflow. You can then define the necessary values for sending and receiving data using the method.

The following XML API methods are exposed by each control. The methods are listed in the order that appear in java file that defines the controls.

Customer Management API

This XML API requires MetaSolv Solution 6.0.3 or higher.

- importCustomerAccount
- getCustomerAccountByKey
- deleteCustomerRequest

Order Management API

This XML API requires MetaSolv Solution 6.0.3 or higher.

- queryOrderManagementRequest
- startOrderByKeyRequest
- updateOrderManagementRequest
- getOrderByKeyRequest
- createOrderByValueRequest
- assignProvisionPlanProcedureRequest
- getActivationDataByKeyRequest
- transferTaskRequest (M6.0.7+)
- updateE911DataRequest (M6.0.6+)
- getE911DataRequest (M6.0.6+)
- updateEstimationCompletedDateRequest (M6.0.8+)
- addTaskJeopardyRequest (M6.0.8+)

- getTaskDetailRequest (M6.0.8+)
- taskJeopardyRequest (M6.0.8+)
- getPSROrderByTN (M6.0.8+)
- processSuppOrder (M6.0.8+)
- getCNAMDataRequest (M6.0.7+)
- getLIBDDataRequest (M6.0.7+)
- updateCNAMDataRequest (M6.0.7+)
- updateLIBDDataRequest (M6.0.7+)
- reopenTaskRequest
- createAttachment
- createOrderRelationshipRequest (M6.0.8+)
- processBillingTelephoneNumber

Inventory Management API

This XML API requires MetaSolv Solution 6.0.3 or higher.

- createEntityByValueRequest
- getServiceRequestDLRsValue
- getEntityByKeyRequest
- updateEntityByValueRequest
- queryInventoryManagementRequest
- updateTNRequest (M6.0.8+)
- tnRecall (M6.0.8+)
- tnValidationRequest
- auditTrailRecording
- getNetworkAreasByGeoAreaRequest
- getNetworkComponentRequest
- getIpAddressesRequest
- inventoryAssociationRequest
- createNewInventoryItemRequest
- queryNetworkLocation (M6.0.14+)
- queryEndUserLocation (M6.0.14+)
- getLocationRequest (M6.0.14+)
- deleteLocationRequest (M6.0.14+)
- updateLocationRequest (M6.0.14+)
- createLocationRequest (M6.0.14)

Network Resource Management API

This XML API requires MetaSolv Solution 6.0.11 or higher.

- getAvailablePhysicalPortsRequest

Service Order Activation API

This XML API requires MetaSolv Solution 6.0.8 or higher.

- createSOAMessageRequest
- getSoaTnsForOrderRequest
- getSoaDefaultsRequest
- getSoaInformationRequest
- getSoaMessageToSendRequest
- setTnSoaCompleteRequest

LSR Management API

This XML API requires LSR 6.10 or higher, LSR 9.2 or higher, or LSR 10.0 or higher.

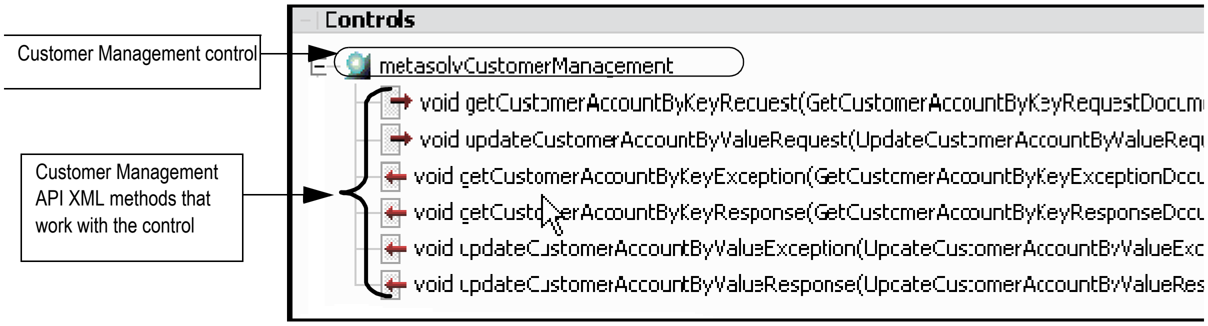
- getLRByKeyRequest
- getDLByKeyRequest
- getLSRByKeyRequest
- getLSRCMByKeyRequest
- createDSCNByValueRequest
- createDSREDByValueRequest
- createLRByValueRequest
- createLSRCMByValueRequest
- createNPLSRByValueRequest
- queryCCNARequest
- queryLSRRequest
- queryLSRForPONCCNAVERRequest
- queryPONSForCCNARequest
- createLSROrderByValueRequest (6.0.8+)

Event Management API

- updateInboundEventStatus
- getEventStatus
- updateOutboundEventStatus

For more information on the control interfaces, see “Appendix B: Control interfaces” on page 89.

The following figure shows how a control and the methods that the control exposes for the corresponding XML API displays in WebLogic Workshop.


Customer Management control
Customer Management API XML methods that work with the control
Controls

```
metasolvCustomerManagement
    void getCustomerAccountByKeyRequest(GetCustomerAccountByKeyRequestDocum
    void updateCustomerAccountByValueRequest(UpdateCustomerAccountByValueRequ
    void getCustomerAccountByKeyException(GetCustomerAccountByKeyExceptionDocu
    void getCustomerAccountByKeyResponse(GetCustomerAccountByKeyResponseDocu
    void updateCustomerAccountByValueException(UpdateCustomerAccountByValueExc
    void updateCustomerAccountByValueResponse(UpdateCustomerAccountByValueRes
```

Figure 7: How controls display in Workshop

Each control has corresponding XSDs that define the XML format that can be received by MetaSolv Solution and show the data fields.The XSDs are constructed using the MetaSolv Information Model (MIM), a dictionary of terms used to standardize data being imported to or exported from MetaSolv Solution using XML.

For a description of the MetaSolv Solution XML API methods, see “Appendix D: XML API methods” on page 127.

MetaSolv Solution schema

Schema, also known as XSDs, are documents that define how XML must be formatted when it is sent to MetaSolv Solution as data input. Where applicable, it also defines how XML will be formatted when MetaSolv returns data. MetaSolv Solution XSDs contain documentation that explains what values are expected, where a value appears in the MetaSolv Solution user interface (where applicable), and the purpose of fields included in the XSDs.

The schema files are housed in two JAR files:

- MetaSolvSchemas.jar

  This file contains the XSD files that define the MetaSolv Solution schema. This file must be pulled into your workspace, as described in "Adding the MetaSolv Solution controls to Workshop". This file is located in *<INSTALLATION_DIRECTORY/mss_samples/APP-INF/lib>* folder.

- Annotated_XML_API_Schemas.jar

  This file contains the same XSD files as MetaSolvSchemas.jar, but these XSD files include annotations that describe the fields in the schema and their corresponding values. This file is used for reference purpose only. This file is located under the Metasolv Solution Developer Documentation Pack in the MetaSolv Solution 6.0 Developer Reference CD.

The MetaSolv Solution schema files are grouped by function. For example, Customer Management, Order Management, Inventory Management, etc. Each functional group defines three separate XSD files. For example, the Customer Management XML API defines the following three files:

- XmlMetaSolvCustomerManagementAPI.xsd
- XmlMetaSolvCustomerManagementEntities.xsd
- XmlMetaSolvCustomerManagementData.xsd

Similarly, the Order Management XML API defines the following three files:

- XmlMetaSolvOrderManagementAPI.xsd
- XmlMetaSolvOrderManagementEntities.xsd
- XmlMetaSolvOrderManagementData.xsd

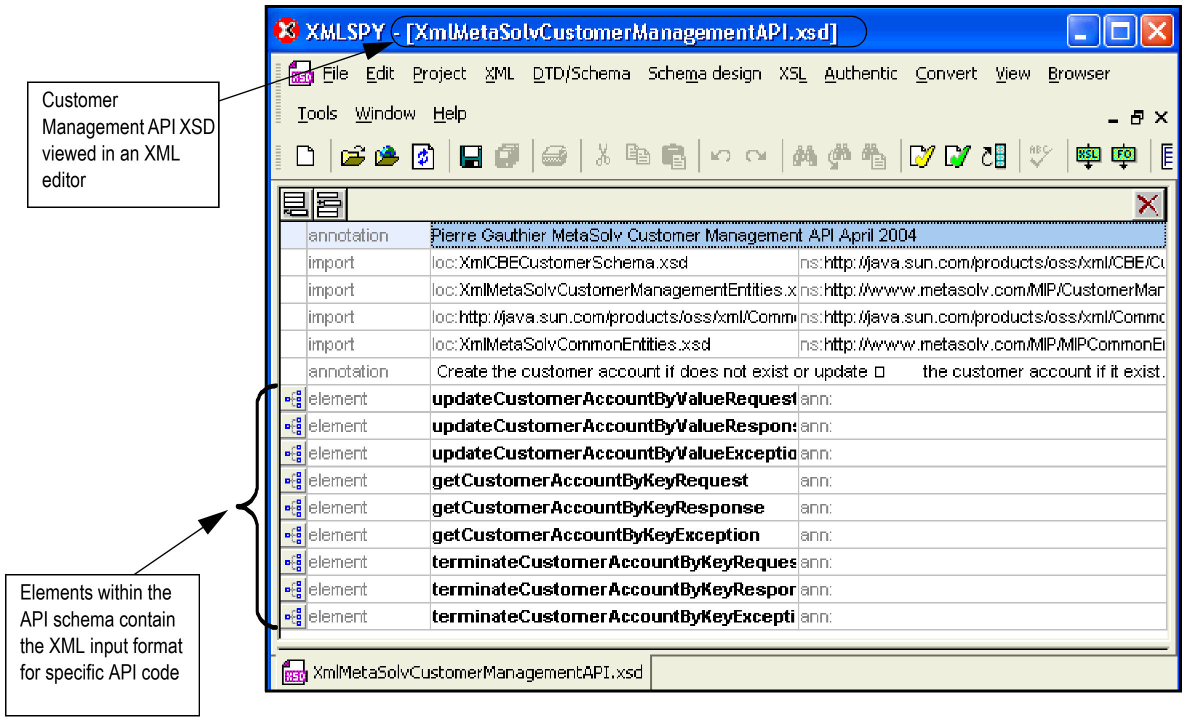
Each of the files define a specific type of information:

- The *API.xsd files define the requests and responses that correspond to the defined control methods covered in the previous section of this chapter.
- The *Entities.xsd files define the data structures that are input to requests or output from responses.

- The *Data.xsd files define various data structures that group data together so the data can be referenced as a complex group, rather than by each individual data element. These data structures are commonly referenced from within an entity data structure. Think of the data structures defined in these files as sub-structures; specifically, the data structures defined in these files are not input to requests or output from responses.

For detailed information on navigating the XSD files, refer to "Appendix C: Navigating the XSD".

The following figure shows the Customer Management schema displayed in an XML editor. Notice the listing of elements in the window.


XMLSPY - [XmlMetaSolvCustomerManagementAPI.xsd]
File Edit Project XML DTD/Schema Schema design XSL Authentic Convert View Browser
Tools Window Help
XmlMetaSolvCustomerManagementAPI.xsd
Customer Management API XSD viewed in an XML editor
Elements within the API schema contain the XML input format for specific API code

annotation	Pierre Gauthier MetaSolv Customer Management API April 2004	
import	loc:XmlCBECustomerSchema.xsd	ns:http://java.sun.com/products/oss/xml/CBE/Cu
import	loc:XmlMetaSolvCustomerManagementEntities.x	ns:http://www.metasolv.com/MIP/CustomerMar
import	loc:http://java.sun.com/products/oss/xml/Comm	ns:http://java.sun.com/products/oss/xml/Commc
import	loc:XmlMetaSolvCommonEntities.xsd	ns:http://www.metasolv.com/MIP/MIPCommonE
annotation	Create the customer account if does not exist or update □	the customer account if it exist.
element	**updateCustomerAccountByValueRequest**	ann:
element	**updateCustomerAccountByValueRespons**	ann:
element	**updateCustomerAccountByValueExceptio**	ann:
element	**getCustomerAccountByKeyRequest**	ann:
element	**getCustomerAccountByKeyResponse**	ann:
element	**getCustomerAccountByKeyException**	ann:
element	**terminateCustomerAccountByKeyReques**	ann:
element	**terminateCustomerAccountByKeyRespor**	ann:
element	**terminateCustomerAccountByKeyExcepti**	ann:

Figure 8: Customer Management API methods displayed in an XML editor (XMLSpy)

The following figure shows *getCustomerAccountByKeyRequest* opened and displayed on the screen.

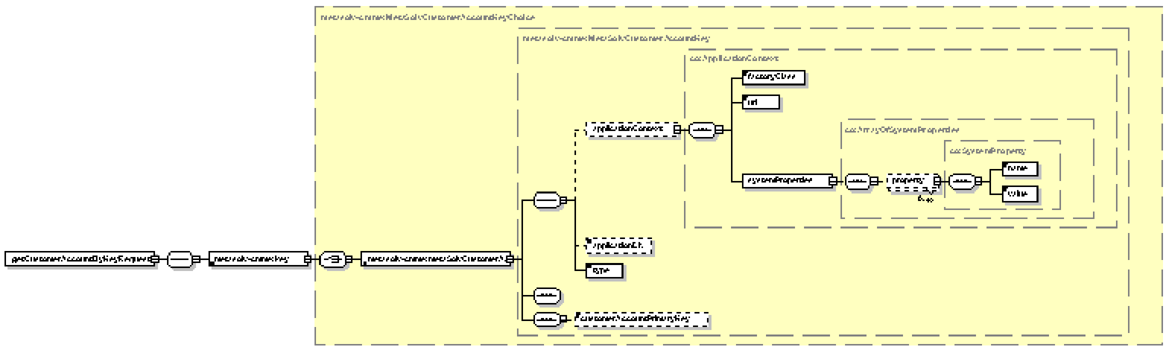
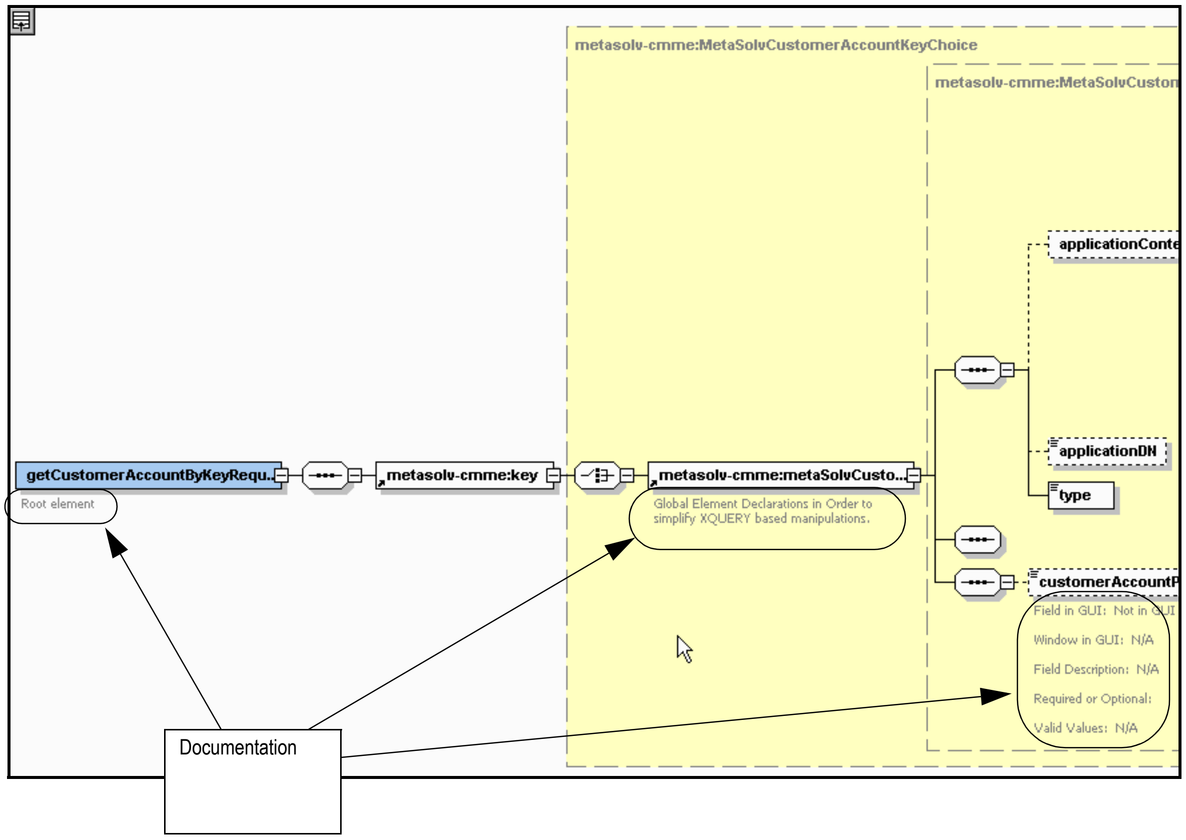


Figure 9: Graphical view of the schema for a method in XMLSpy

The schema include documentation on the information they contain. The following figure shows the graphical representation of an element's schema with documentation highlighted.


metasolv-cmme:MetaSolvCustomerAccountKeyChoice
metasolv-cmme:MetaSolvCuston
applicationConte
applicationDN
type
customerAccountP
getCustomerAccountByKeyRequ..
Root element
metasolv-cmme:key
metasolv-cmme:metaSolvCusto...
Documentation

Global Element Declarations in Order to simplify XQUERY based manipulations.

Field in GUI: Not in GUI
Window in GUI: N/A
Field Description: N/A
Required or Optional:
Valid Values: N/A

Figure 10: Schema documentation shown in a graphical view in XMLSpy

WebLogic Platform 8.1

WebLogic Platform 8.1 SP5 provides the environment for the MetaSolv Solution integration process. WebLogic Platform is a powerful software application that has many uses beyond the scope of this document. This document documents only those portions of WebLogic Platform and its software components that relate directly to accomplishing a task in the integration of MetaSolv Solution. BEA Software provides a large body of information on WebLogic Platform at http://e-docs.bea.com.

WebLogic Platform contains a full-featured integrated development environment (IDE) that you can use to create and debug your application. WebLogic Workshop provides the tools to automate much of the coding that is required for development. The following figure shows a Workshop workflow, which is the graphical representation of steps that combine methods and data transformations to accomplish a task in MetaSolv Solution.

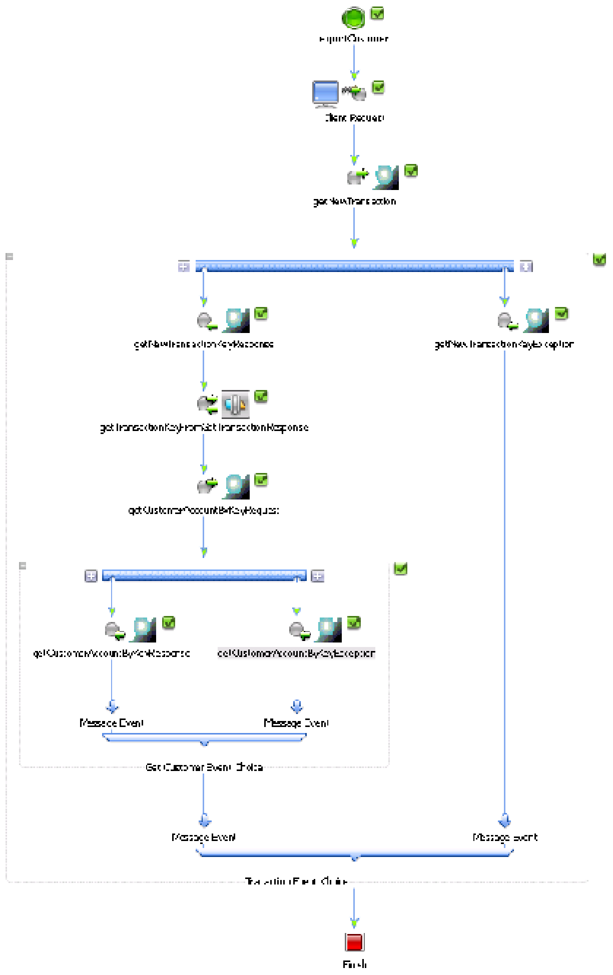

ExportCustomer
Client Request
getNewTransaction
getNewTransactionKeyResponse
getNewTransactionKeyException
getTransactionKeyFromGetTransactionResponse
getCustomerAccountByKeyRequest
getCustomerAccountByKeyResponse
getCustomerAccountByKeyException
Message Event
Message Event
Get Customer Event Choice
Message Event
Message Event
Transaction Event Choice
Finish

Figure 11: A WebLogic Workshop workflow for exporting customer information

To integrate MetaSolv Solution with another application, you must use the following WebLogic Platform components:

- **WebLogic Workshop**

  This component provides an integrated development, deployment, and run-time environment for building applications. The procedures for developing a workflow and the MetaSolv Solution samples will be described using Workshop in this document.

- **WebLogic Integration**

  This component provides a framework for developing and integrating applications and business processes from within and across an enterprise.

Basic integration steps

The following steps show the high level process for integrating MetaSolv Solution with a third-party application. Some steps can be performed in a different order, but the order shown here is recommended by MetaSolv as a best practice.

1. Layout the steps in your project to determine any data mapping that must be done between your schemas and the MetaSolv Solution schemas.

   This means identifying the nodes, or building blocks, in what will become the workflow for the application.

2. Identify the sources and targets for all data mapping that you identify between the schemas.

   This step includes both requests into MetaSolv Solution and the responses that are returned to your external system.

3. Build the transformations in WebLogic Workshop.
4. Build the workflow for the application in WebLogic Workshop.
5. Build the framework.

   This step is optional. It includes setting up logging and error handling.

6. Test the workflow.

   Some basic test capabilities are included in WebLogic Workshop. You can also use the connector included in the mss_samples.jar file to simulate receiving and processing data from an external application in WebLogic Workshop.

7. Create an ear file that contains the application and deploy it with MetaSolv Solution.

Special characters

The XML API supports the special characters listed in the table below. The first five rows show special characters that are recognized by the API as an entity other than the special character itself. The remaining rows list special characters that are recognized by the API in the same manner as the GUI.

Table 1: Special characters supported by the XML API

Special character	API	GUI
’	&apos;	’
"	&quot;	The " must be last character.
&	&amp;	&
<	&lt;	<
>	&gt;	>
~	~	~
!	!	!
@	@	@
#	#	#
$	$	$
%	%	%
^	^	^
*	*	*
(	(	(
)	)	)
{	{	{
}	}	}
[	[	[
]	]	]

3

Developing an integration application

This chapter provides information to help get you started developing applications using the MetaSolv Integration and Portal Toolkit. The information indicates through a simple example how to use the tools in the toolkit. When you understand the tools, and you have a clear determination of what you want to accomplish, you are ready to begin developing applications.

You will be shown how to accomplish the major steps in creating and deploying a MetaSolv Solution integration application using the *GetCustomer* sample included in *mss_samples.jar*. The sections in this chapter follow the basic steps for creating an application for MetaSolv Solution. To create an application for MetaSolv Solution in WebLogic Workshop, you must:

1. Plan the application.
2. Create a new (empty) application in WebLogic Workshop.
3. Import the appropriate MetaSolv Solution controls into the new workflow in Workshop.
4. Create data transformation controls for your incoming XML data and for the outgoing data you expect to receive in response.
5. Create the workflow using MetaSolv Solution controls, the transformation controls you created, and generic controls inside Workshop.
6. Set up logging and exception handling for the application.
7. Test the workflow.
8. Create and deploy the application .ear file.

All of the steps listed are explained in the following sections. The sections describe the process for creating the *GetCustomer* example included in *mss_samples.jar*. For information on how to locate the samples file, see "Appendix A: XML API sample code" on page 81. The *GetCustomer* example is simple and easy to understand, but it contains processes that illustrate how controls are used to integrate MetaSolv Solution with another system.

The input for the *GetCustomer* example comes from the *Cim_customer.xsd* also included in Samples.jar. The details of the example in this chapter may vary slightly from the *GetCustomer* example in the *mss_samples.jar* file. When that is true, it is done to simplify the example in this chapter for illustration purposes.

This figure shows where the GetCustomer .jpd file (getCustomerHttpSample) is located inside the mss_samples directory. For more information on the samples in the directory, see "Appendix A: XML API sample code" on page 81.

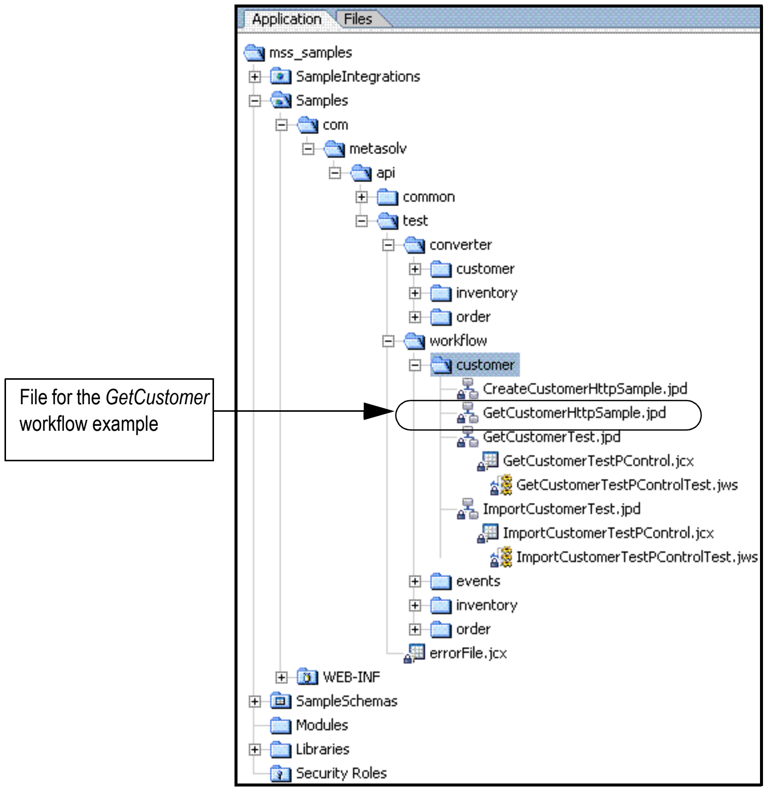


Figure 12: Directory structure for mss_samples

When you open GetCustomerHttpSample in Workshop, the following workflow appears on the Workshop canvas.

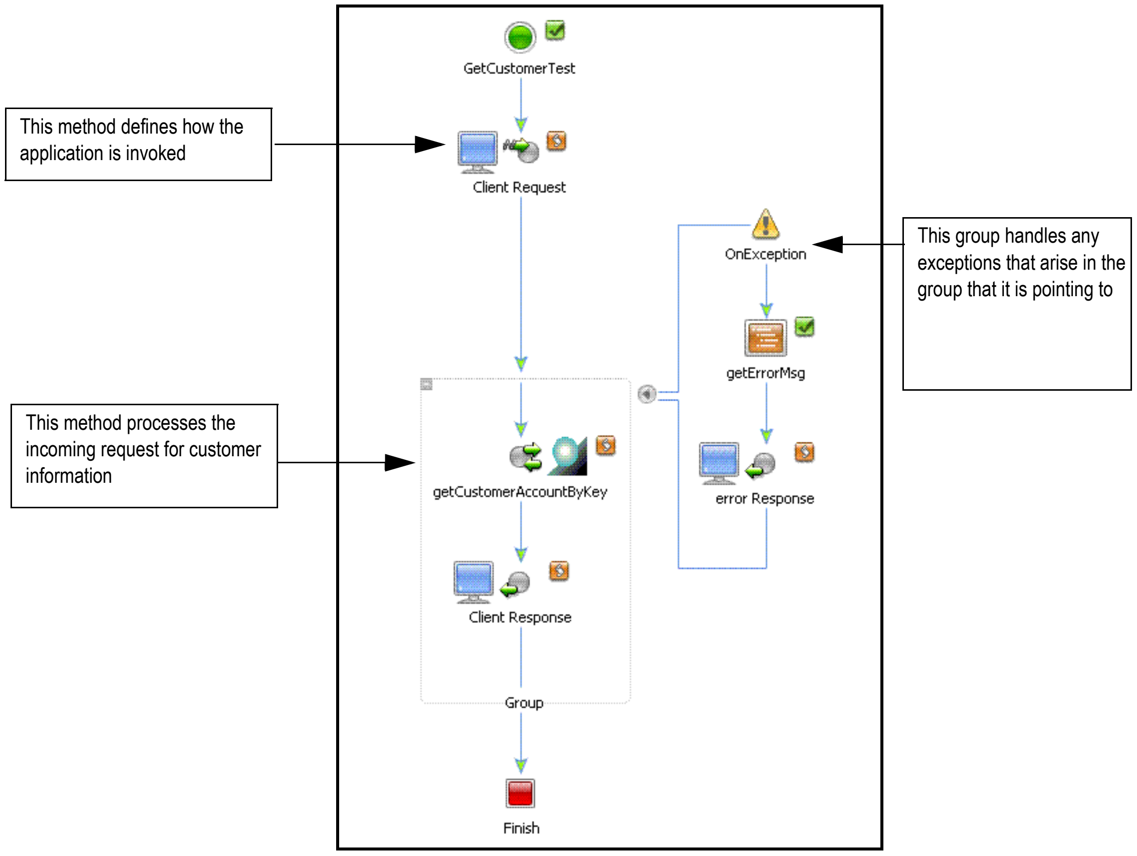

This method defines how the application is invoked
This method processes the incoming request for customer information
This group handles any exceptions that arise in the group that it is pointing to
GetCustomerTest
Client Request
OnException
getErrorMsg
error Response
getCustomerAccountByKey
Client Response
Group
Finish

Figure 13: Workflow for getCustomerHttpSample.jpd

The following figure shows a graphical view of the schema for Cim_Customer, which provides the format for the incoming XML for the *GetCustomer* example shown in this chapter.

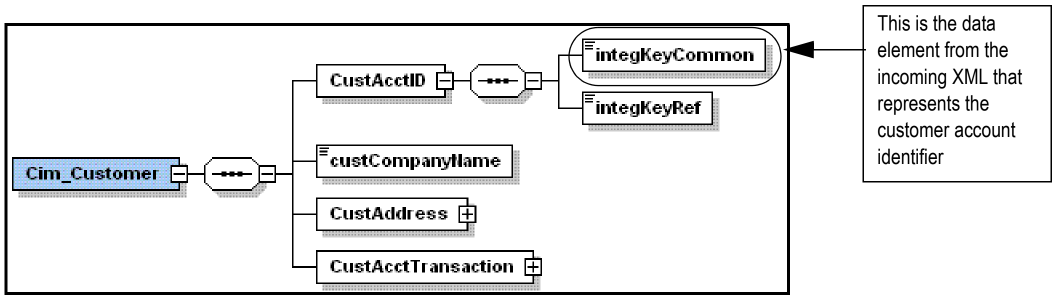

This is the data element from the incoming XML that represents the customer account identifier
Cim_Customer
CustAcctID
integKeyCommon
integKeyRef
custCompanyName
CustAddress
CustAcctTransaction

Figure 14: Schema for the incoming request XML

Planning the application

This section describes how to plan for your application. MetaSolv recommends as a best practice that you list out the information you need before you begin work on the application in Workshop. The information you need includes:

- The BEA domain to which the application will be assigned in Workshop. You can use an existing domain or create a new domain.
- MetaSolv Solution controls you will require.
- A list of the data that needs to be transformed from your XML format to the MetaSolv Solution MIM format.

Here is the information needed for the GetCustomer example used in this chapter.

MetaSolv Solution controls required:

MetaSolv Customer Management API

Transformations required:

Incoming XML: *customer account identifier*

Outgoing XML: *customer account identifier, customer's company name*

The output was limited to two data items to keep the example simple. Although this example uses simple inputs and outputs, the transformations for a normal integration effort can become complex and therefore requires careful planning. The problem of knowing which data from your XML maps into which data in the MetaSolv Solution schema is eased by the documentation in the MetaSolv Solution schema. See "MetaSolv Solution schema" on page 14 for more information about the MetaSolv Solution schemas.

Creating a new application in Workshop

To create a new application

1. Start WebLogic Workshop by doing one of the following:

   Windows:

   From the Start menu, select **Start>Programs>BEA WebLogic Platform 8.1>WebLogic Workshop 8.1**.

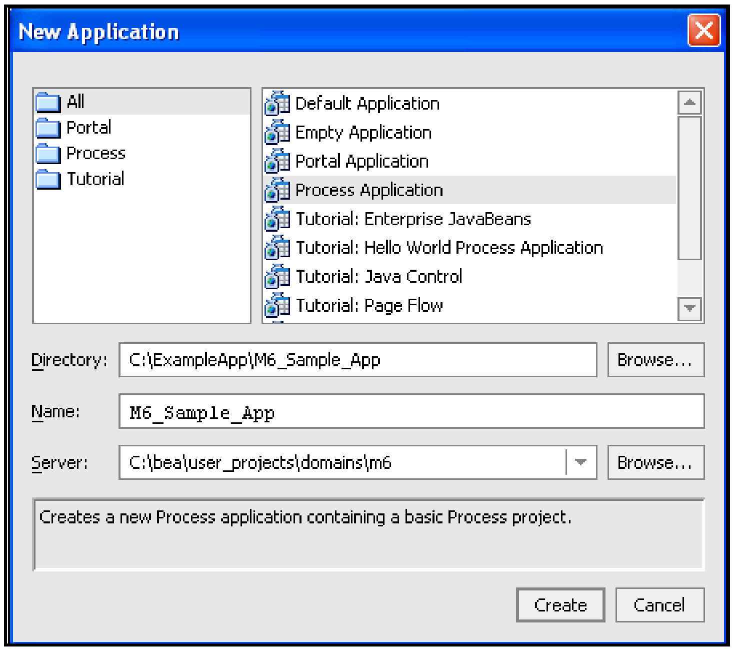
   UNIX:

   $BEA_HOME/weblogic81/workshop/Workshop.sh.

   If this is the first time starting WebLogic Workshop, or if Workshop was previously open on an application from another WebLogic domain, you might see a warning about fixing the domain. If so, click **Continue** to ignore the warning, then close the open application.

2. Select **File>New>Application**.

The New Application window appears.



Directory: C:\ExampleApp\M6_Sample_App

Name: M6_Sample_App

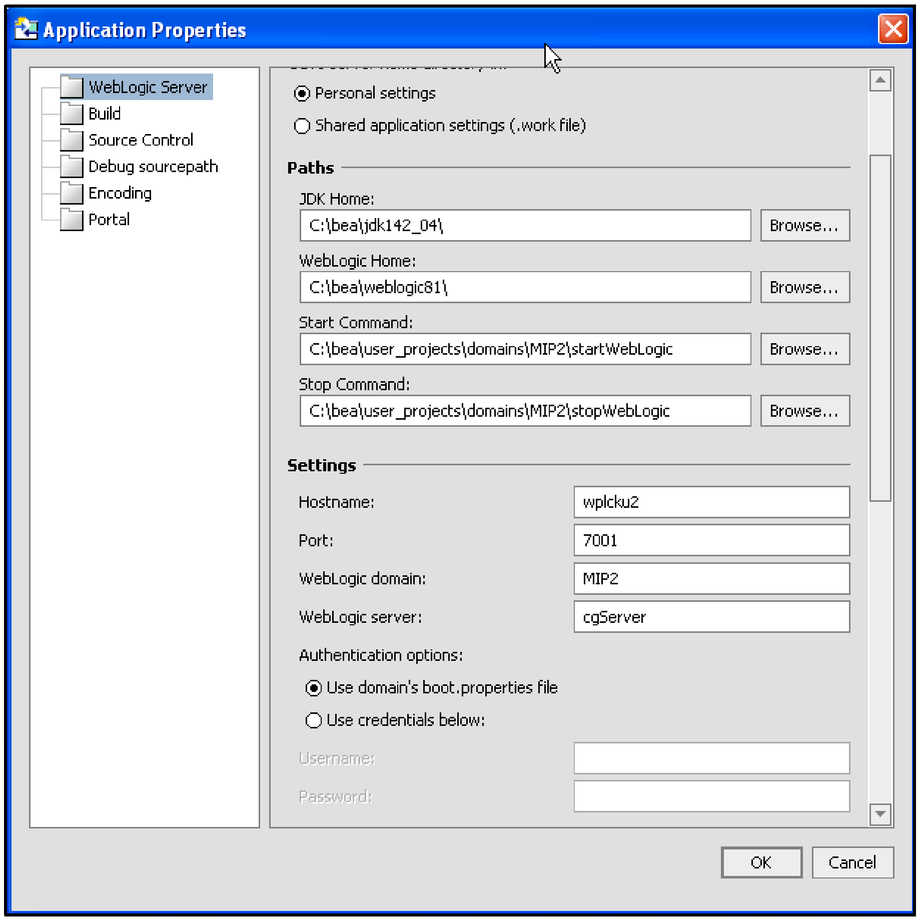
Server: C:\bea\user_projects\domains\m6

Creates a new Process application containing a basic Process project.

3. Complete the following information on the window and click **Create**:
   a. Select **Process Application** in the list box.
   b. Accept the default directory location for the application or click **Browse** to select another directory.
   c. Type a name for the new application in the **Name** field (for example, M6_Sample_App).
   d. Select the domain directory in the **Server** field.

   When you click Create, the new application is created in Workshop.
4. In the Workshop menu bar, select **Tools**>**Application Properties.**

The Application Properties window appears.



[✓ Personal settings] Shared application settings (.work file)

Paths

JDK Home: C:\bea\jdk142_04\

WebLogic Home: C:\bea\weblogic81\

Start Command: C:\bea\user_projects\domains\MIP2\startWebLogic

Stop Command: C:\bea\user_projects\domains\MIP2\stopWebLogic

Settings

Hostname:	wplcku2
Port:	7001
WebLogic domain:	MIP2
WebLogic server:	cgServer

Authentication options:

[✓ Use domain's boot.properties file] Use credentials below:

Username:

Password:

5. Complete the following tasks on the Application Properties window and click **OK**:
   - Check to make sure the server name directory is the domain directory.
   - In the Settings section of the Application properties, change the Hostname from *localhost* to the name of the machine running Workshop.

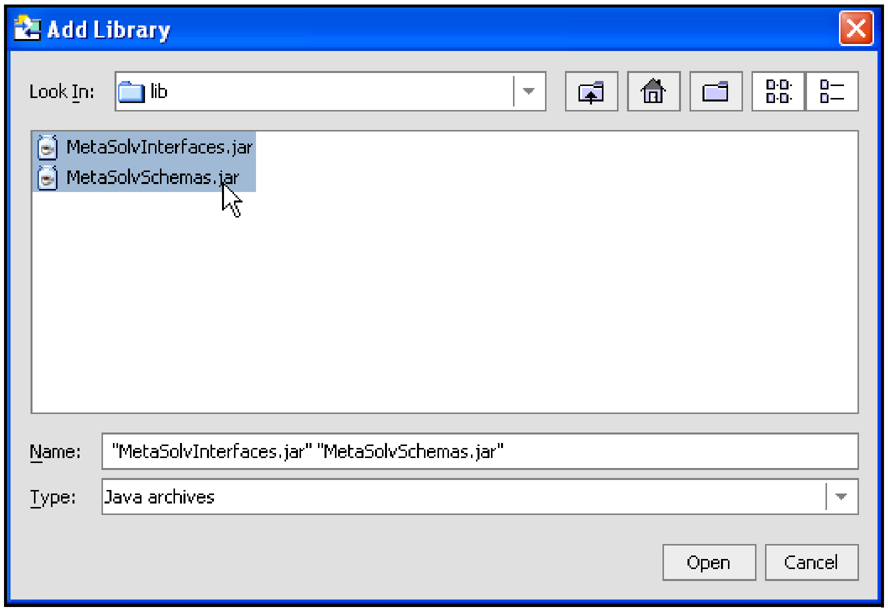
When the connection is made to the adapter's domain server, a green light shows on the status bar with a note indicating the server is running.

Adding the MetaSolv Solution controls to Workshop

To add the MetaSolv Solution controls

1. In the left pane treeview, right-click **Libraries**, then click **Add Libraries**.

   The Add Library dialog box appears for the selection of the files whose contents can be imported into Workshop

   Name: "MetaSolvInterfaces.jar" "MetaSolvSchemas.jar"
   Type: Java archives

2. In the dialog box, navigate to the directory that contains the MetaSolv Solution controls.

   For example, the controls are included in the mss_samples.jar file. Locate the directory where you unjarred the mss_samples file and look for the following directory:

   <*INSTALLATION_DIRECTORY/mss_samples*>/APP-INF/lib

3. Select MetaSolvInterfaces.jar and MetaSolvSchemas.jar, and click **Open**.

   This action makes the MetaSolv Solution controls available to be used in Workshop.

Creating data transformations

The next step in creating an application is the creation of controls for transforming data. The GetCustomer example has data transformations for the request data (incoming) and the response data (outgoing). A transformation file for each set of data must be created. the following figure shows a simple example of the process.

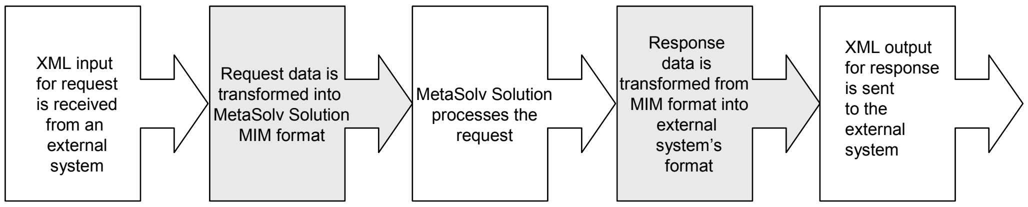

XML input for request is received from an external system
Request data is transformed into MetaSolv Solution MIM format
MetaSolv Solution processes the request
Response data is transformed from MIM format into external system's format
XML output for response is sent to the external system

Figure 15: Transforming XML input and output files

The following procedure shows the steps used to create a transformation file for the Cim_Customer example XSD.

Request transformation control

To create a transformation file for the incoming data

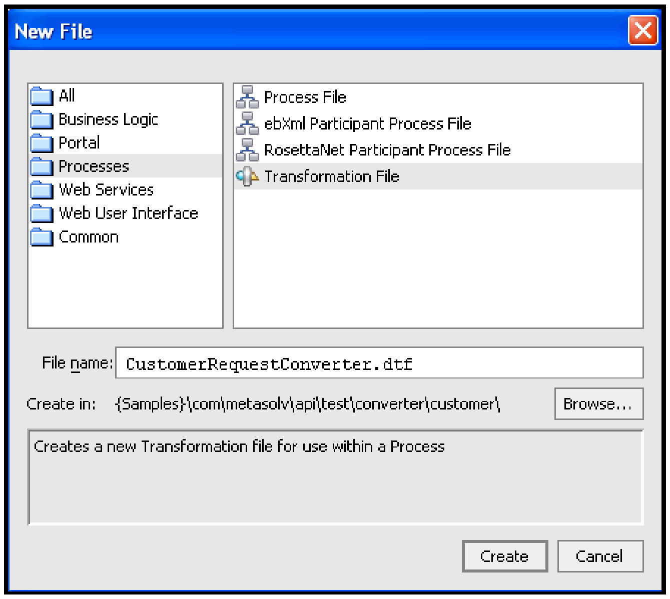
1. Select a location for the file in the treeview on the left pane.

   You can create a directory or place the file in an existing directory.

2. In the treeview, right-click on the directory where the transformation file is to reside and select **New>Transformation file**.

   For example, in the GetCustomer example, the file will reside in the following directory: sample/com/metasolv/api/test/converter. See the figure on page 22 for information on the directory structure.

   The following New File dialog box appears.



File name: CustomerRequestConverter.dtf

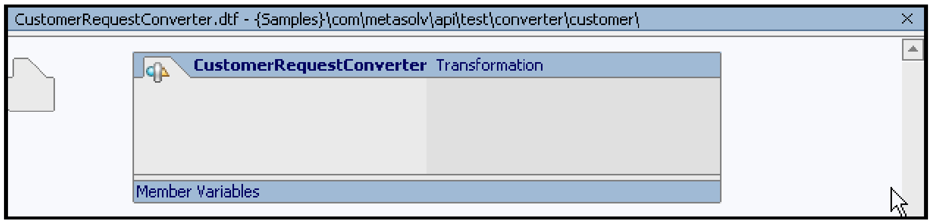
Create in: {Samples}\com\metasolv\api\test\converter\customer\

Creates a new Transformation file for use within a Process

3. In the dialog box, complete the following actions:
   a. Select **Processes** in the list on the left, and **Transformation file** in the list on the right.
   b. Type a name for the file that indicates it purpose.

      For example, *CustomerRequestConverter.dtf*.
   c. Accept the default location for the file or click **Browse** to find another directory location.
   d. Click **Create**.

   The following Transformation window appears.

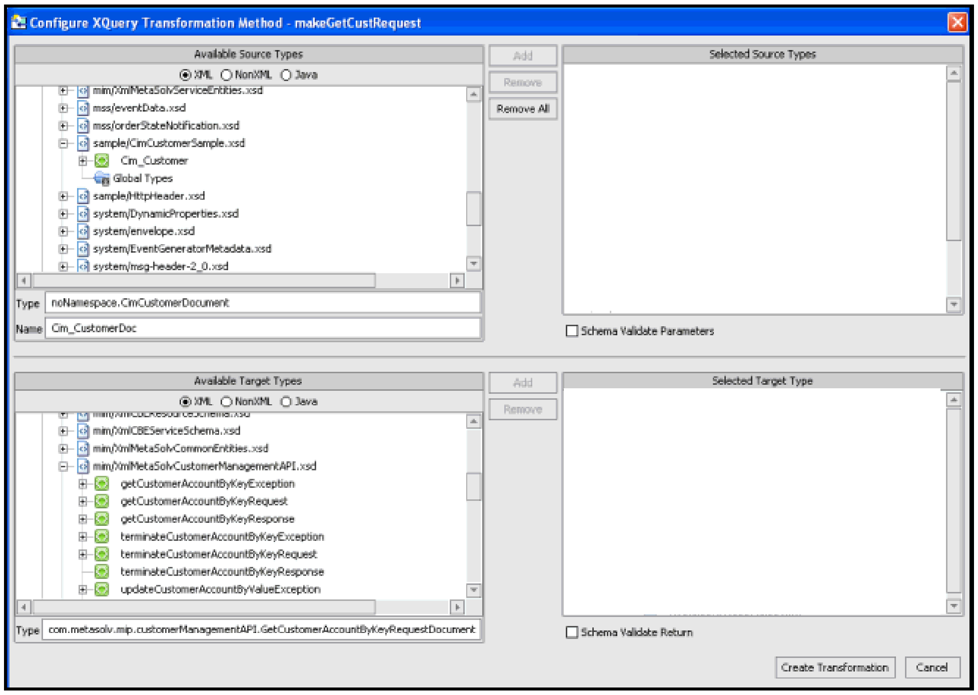

CustomerRequestConverter.dtf - {Samples}\com\metasolv\api\test\converter\customer\
CustomerRequestConverter Transformation
Member Variables

4. Right-click in the transformation window and select **Add transformation method** from the popup menu that appears.

   A new method appears listed in the left side of the window.
5. Type the method name in the text field below the method, where the cursor appears.

   For example, *makeGetCustRequest*.
6. Right-click on the new method, and select **Configure XQuery transformation method**.

The Configure XQuery Transformation Method window appears.



Available Source Types: [✓ XML] NonXML Java

mim/XmlMetaSolvServiceEntities.xsd
mss/eventData.xsd
mss/orderStateNotification.xsd
sample/CimCustomerSample.xsd
Cim_Customer
Global Types
sample/HttpHeader.xsd
system/DynamicProperties.xsd
system/envelope.xsd
system/EventGeneratorMetadata.xsd
system/msg-header-2_0.xsd

Type: noNamespace.CimCustomerDocument
Name: Cim_CustomerDoc

Available Target Types: [✓ XML] NonXML Java

mim/XmlCBEServiceSchema.xsd
mim/XmlMetaSolvCommonEntities.xsd
mim/XmlMetaSolvCustomerManagementAPI.xsd
getCustomerAccountByKeyException
getCustomerAccountByKeyRequest
getCustomerAccountByKeyResponse
terminateCustomerAccountByKeyException
terminateCustomerAccountByKeyRequest
terminateCustomerAccountByKeyResponse
updateCustomerAccountByValueException

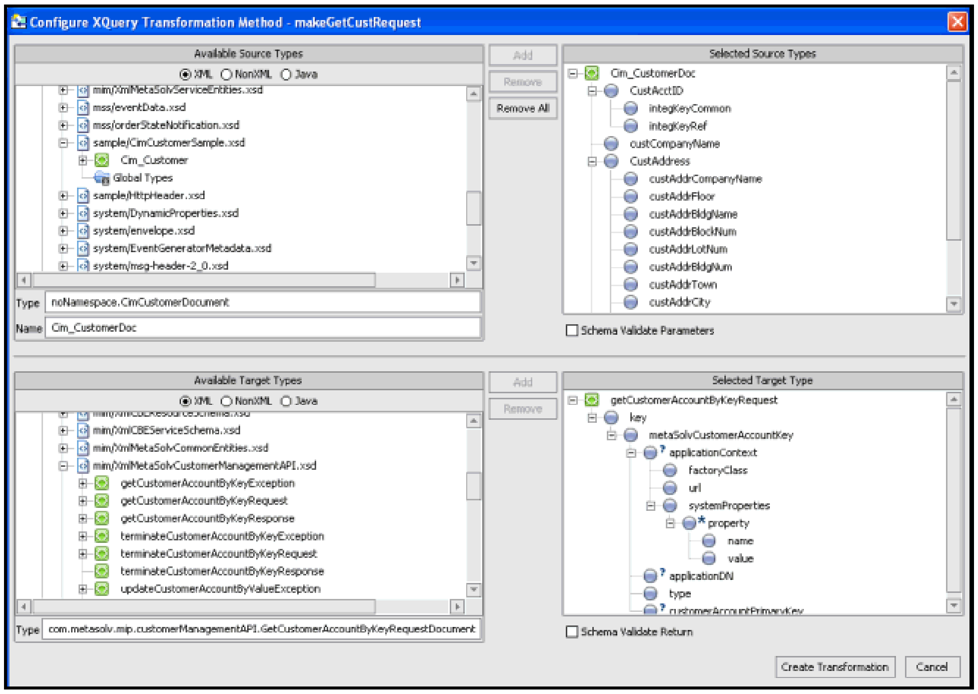
Type: com.metasolv.mip.customerManagementAPI.GetCustomerAccountByKeyRequestDocument

This window contains four panes that allow you to define the source and the target XML transformation files.

7. In the Available Source Types pane, select the **XML** option as the source type, then locate the schema for the XML input that will be received by the integration layer, and click **Add** to display the schema on the right side.

8. In the Available Target Types pane, select the **XML** option as the source type, then locate the schema for the MIM XML format you want to transform the input into, and click **Add** to display the schema on the right side.

The following figure shows the window with the schemas displayed.



Configure XQuery Transformation Method - makeGetCustRequest

Available Source Types: (●) XML ( ) NonXML ( ) Java

```
mim/XmlMetaSolvServiceEntities.xsd
mss/eventData.xsd
mss/orderStateNotification.xsd
sample/CimCustomerSample.xsd
    Cim_Customer
    Global Types
sample/HttpHeader.xsd
system/DynamicProperties.xsd
system/envelope.xsd
system/EventGeneratorMetadata.xsd
system/msg-header-2_0.xsd
```

Type: noNamespace.CimCustomerDocument

Name: Cim_CustomerDoc

Selected Source Types:

```
Cim_CustomerDoc
    CustAcctID
        integKeyCommon
        integKeyRef
    custCompanyName
    CustAddress
        custAddrCompanyName
        custAddrFloor
        custAddrBldgName
        custAddrBlockNum
        custAddrLotNum
        custAddrBldgNum
        custAddrTown
        custAddrCity
```

[ ] Schema Validate Parameters

Available Target Types: (●) XML ( ) NonXML ( ) Java

```
mim/XmlCBEServiceSchema.xsd
mim/XmlMetaSolvCommonEntities.xsd
mim/XmlMetaSolvCustomerManagementAPI.xsd
    getCustomerAccountByKeyException
    getCustomerAccountByKeyRequest
    getCustomerAccountByKeyResponse
    terminateCustomerAccountByKeyException
    terminateCustomerAccountByKeyRequest
    terminateCustomerAccountByKeyResponse
    updateCustomerAccountByValueException
```

Type: com.metasolv.mip.customerManagementAPI.GetCustomerAccountByKeyRequestDocument

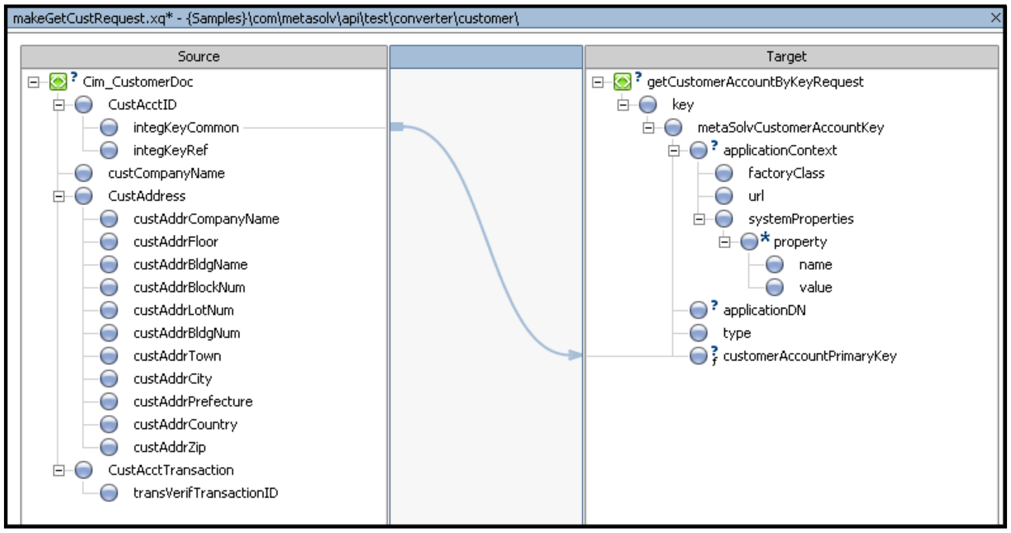
Selected Target Type:

```
getCustomerAccountByKeyRequest
    key
        metaSolvCustomerAccountKey
            ? applicationContext
                factoryClass
                url
                systemProperties
                    * property
                        name
                        value
            ? applicationDN
            type
            ? customerAccountPrimaryKey
```

[ ] Schema Validate Return

9. Click **Create Transformation**.

   The mapping window appears.



makeGetCustRequest.xq* - {Samples}\com\metasolv\api\test\converter\customer\

Source:

```
? Cim_CustomerDoc
    CustAcctID
        integKeyCommon
        integKeyRef
    custCompanyName
    CustAddress
        custAddrCompanyName
        custAddrFloor
        custAddrBldgName
        custAddrBlockNum
        custAddrLotNum
        custAddrBldgNum
        custAddrTown
        custAddrCity
        custAddrPrefecture
        custAddrCountry
        custAddrZip
    CustAcctTransaction
        transVerifTransactionID
```

Target:

```
? getCustomerAccountByKeyRequest
    key
        metaSolvCustomerAccountKey
            ? applicationContext
                factoryClass
                url
                systemProperties
                    * property
                        name
                        value
            ? applicationDN
            type
            ? customerAccountPrimaryKey
```

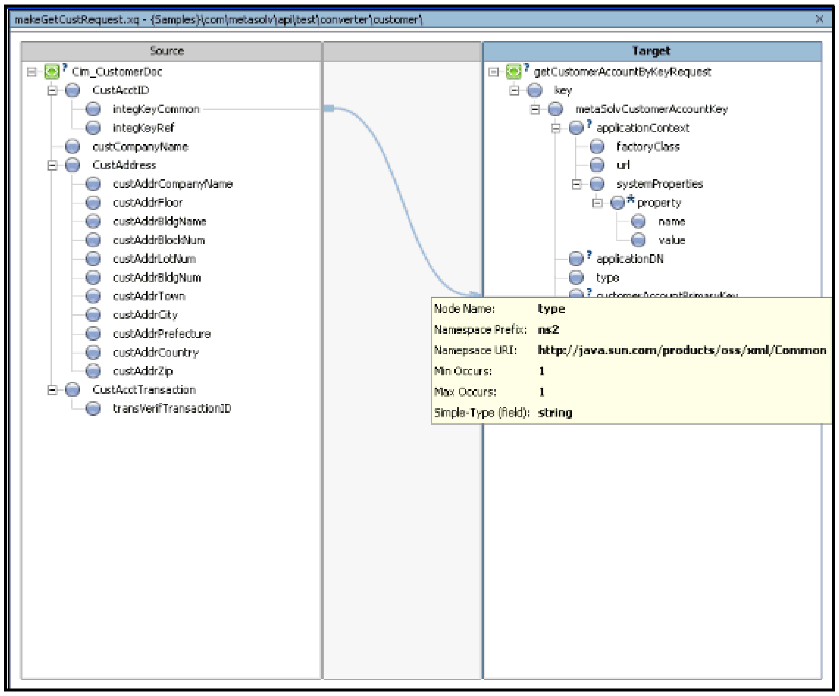
10. Drag the data element to be mapped from the Source list and drop it on the target element you want it mapped to.

    The preceding figure shows the only data element to be mapped for the incoming XML for the *GetCustomer* example. Only the customer account number is required to export customer information in this example.

11. In the Target list, set a value for the data element **type** by right-clicking on the element and selecting **Create constant**. In the dialog box that appears, type **""** in the **Constant value** field and click **OK**.

    Some data elements require a value. If no incoming value is mapped to the data elements from the source XML file, you must create a constant value for the data element. In the target schema shown in the previous figure, the data element **type** requires a value.

    You can hover the cursor over a data element to see documentation on the element and determine whether a value must be assigned. As a best practice, this information should be determined beforehand from the schema using an XML editor. The following figure shows the data element **type** with the documentation for the element.



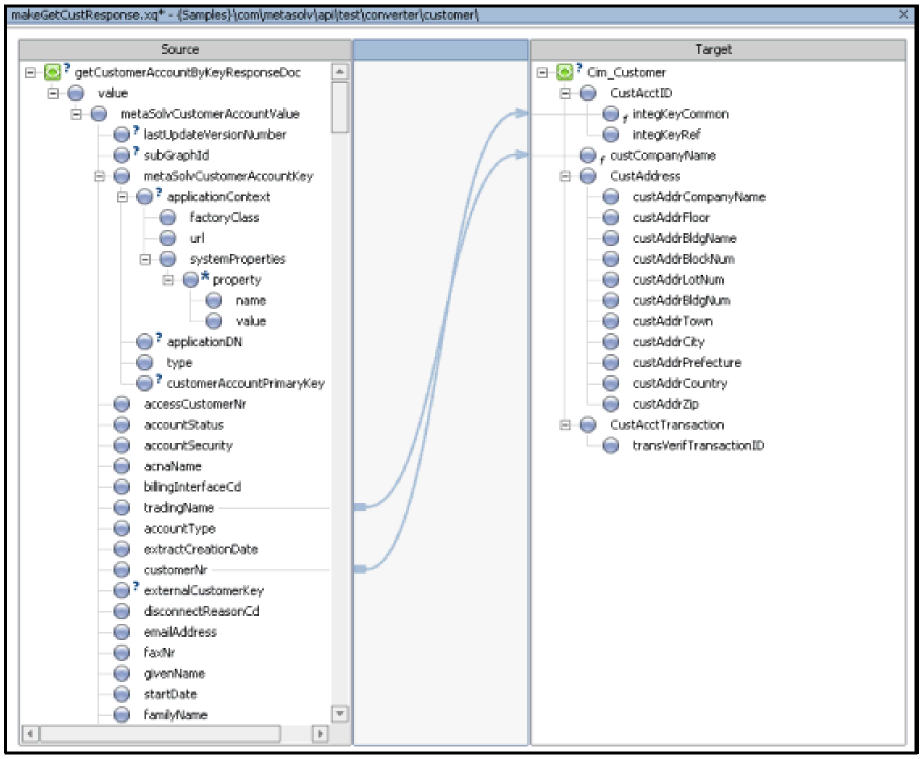
Node Name: type
Namespace Prefix: ns2
Namepsace URI: http://java.sun.com/products/oss/xml/Common
Min Occurs: 1
Max Occurs: 1
Simple-Type (field): string

12. Click **Save** and close the mapping window.

Response transformation control

The response transformation for the *GetCustomer* example is created in the same manner as the request transformation. A new file is created, and the associated method is named *makeGetCustResponse.* The following figure shows the mapping for the response transformation.

Two elements are mapped for the response: *customer account number* and *company name*.



Building the workflow

This section describes how to create a workflow in Workshop. The workflow contains all of the control and transformation methods necessary to complete the integration tasks you require. Workshop gives you the ability to construct the workflow graphically and generate the code automatically.

Step 1: Creating the workflow process file

Each workflow has a .jpd process file. This section explains how to create the process file and the workflow in Workshop.

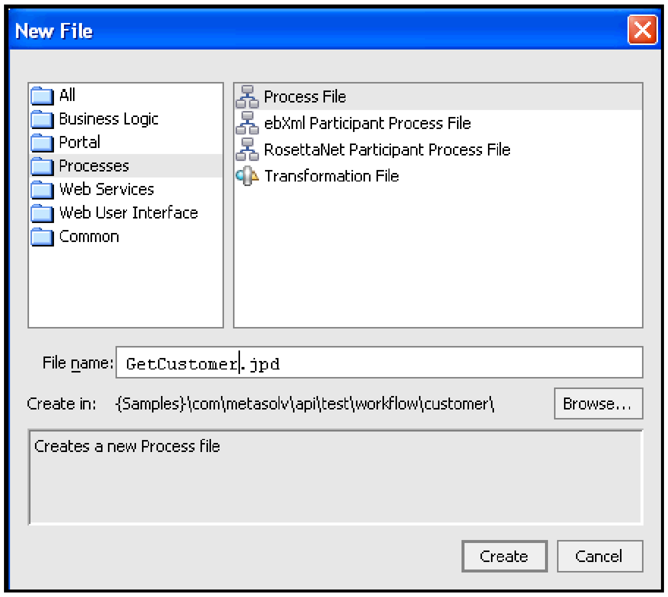
To create a workflow

1. Select a location for the .jpd process file in the treeview on the left pane.

   You can create a directory or place the file in an existing directory.

2. In the treeview, right-click on the directory where the workflow file is to reside and select **New>Process file**.

   The following New File dialog box appears.



File name: GetCustomer.jpd

Create in: {Samples}\com\metasolv\api\test\workflow\customer\

Creates a new Process file

3. In the dialog box, complete the following actions:

   a. Select **Processes** in the list on the left, and **Process file** in the list on the right.

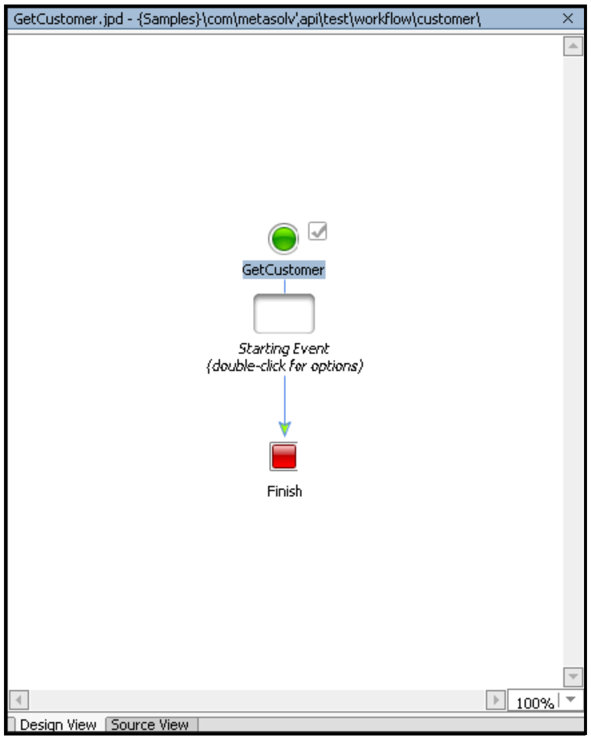
   b. Type a name for the file that indicates it purpose.

      For example, *GetCustomer.jpd*

c. Accept the default location for the file or click **Browse** to find another directory location.

d. Click **Create**.

An empty workflow appears on the Workshop canvas. The following figure shows the new *GetCustomer* workflow.


GetCustomer.jpd - {Samples}\com\metasolv\api\test\workflow\customer\
GetCustomer
Starting Event
(double-click for options)
Finish
100%
Design View
Source View

Once an empty workflow is created, you can add the controls to be used in the workflow to the Data Palette.

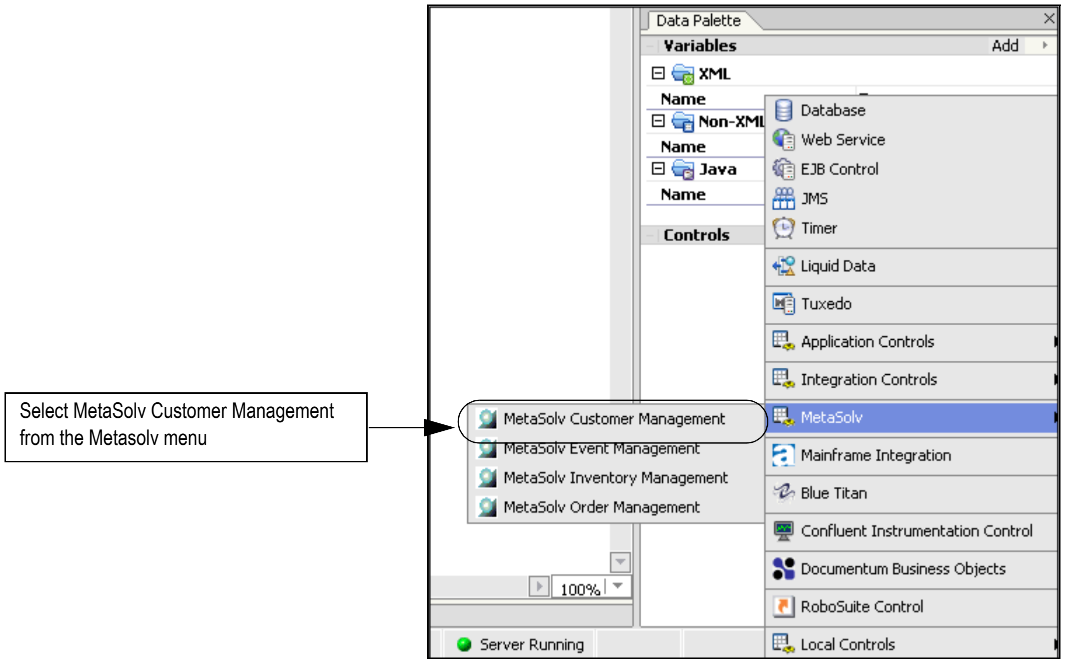
Step 2. Adding controls to the Workshop Data Palette

This section explains how to add controls to the Workshop Data Palette for use in a workflow. The Customer Management control and the data transformation controls shown in "Creating data transformations" on page 28 are added to the Data Palette.

Adding the Customer Management control

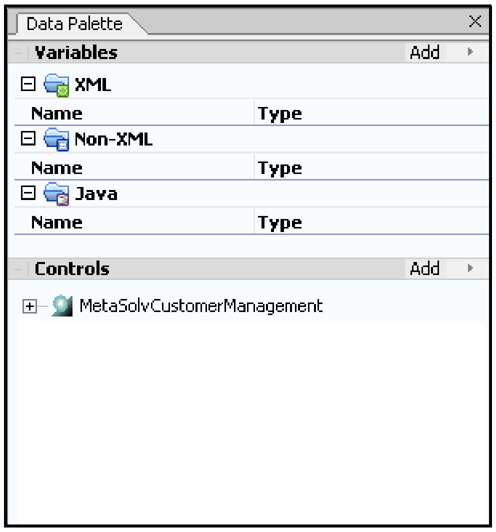
To a control to the data palette

1. In the Controls section of the Data Palette on the right side of the Workshop window, click **Add** and select **MetaSolv Customer Management** from the MetaSolv menu.



2. In the dialog box that appears, type a name for the control and click **OK**.

For example, you can type *MetaSolvCustomerManagement*. The control and its methods appear in the Data Palette pane where they are available for the Workshop canvas.



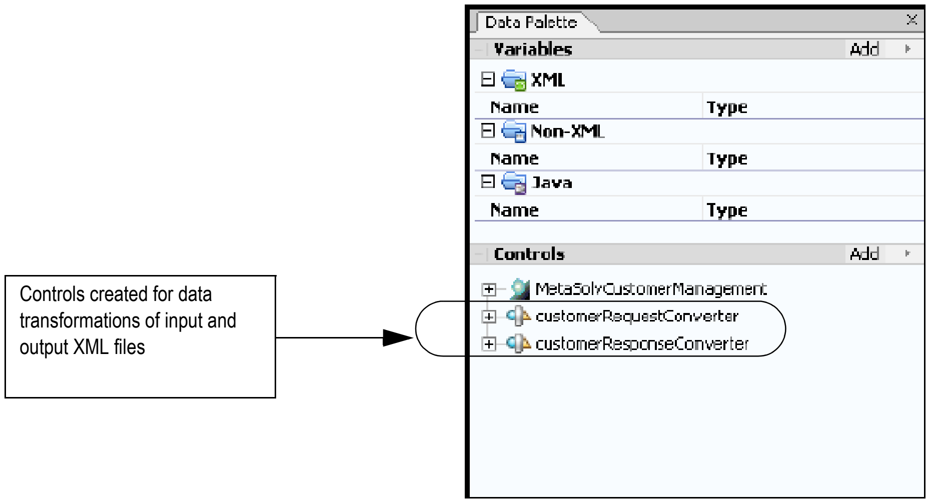
Adding a data transformation control

This section shows how to add *customerRequestConverter* and *customerResponseConverter* to the Data Palette. These controls were built in Workshop to transform data.

To display transformation controls in the Data Palette

- On the left pane treeview, select the .dtf files you created for transformations and drag them to the Control section of the Data Palette and drop them.

The following figure shows the two transformation files created for *GetCustomer* as controls in the Data palette.

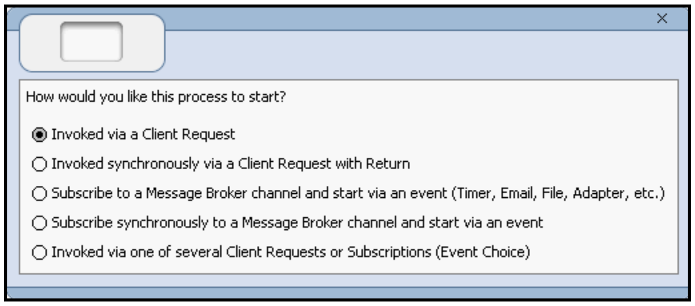

Controls created for data transformations of input and output XML files
Data Palette
Variables
Add
XML
Name
Type
Non-XML
Name
Type
Java
Name
Type
Controls
Add
MetaSolvCustomerManagement
customerRequestConverter
customerResponseConverter

Step 3. Specifying how the request is invoked

To specify how the request is invoked

1. Double-click the Starting Event node.

   The following dialog box appears.



How would you like this process to start?

- [✓ Invoked via a Client Request]
- Invoked synchronously via a Client Request with Return
- Subscribe to a Message Broker channel and start via an event (Timer, Email, File, Adapter, etc.)
- Subscribe synchronously to a Message Broker channel and start via an event
- Invoked via one of several Client Requests or Subscriptions (Event Choice)

2. Select the **Invoked via a Client Request** option and close the dialog box.

   The Client Request node appears in the workflow.

3. Double-click the Client Request node.

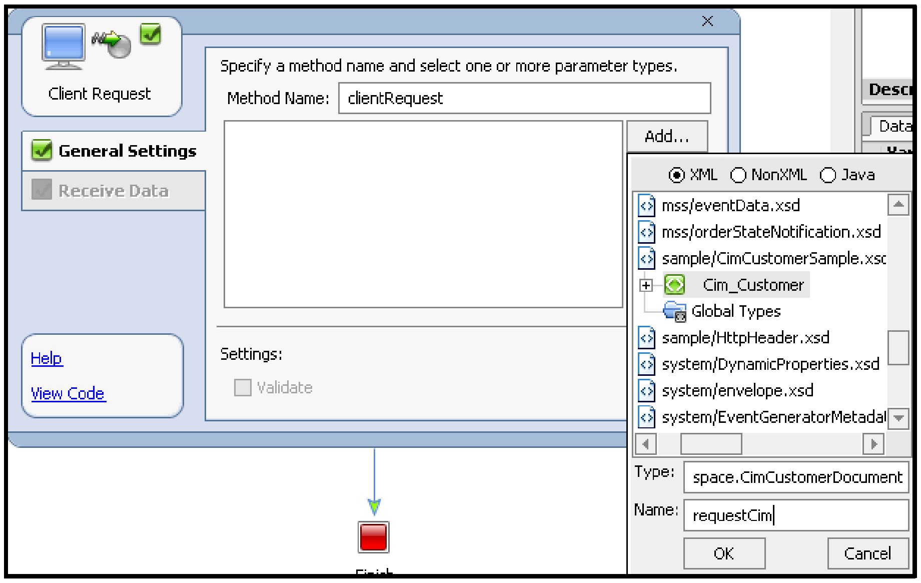
   The Client Request dialog box appears. Double-clicking any node on the Workshop canvas causes its properties dialog box to appear. This dialog box allows you to name the method the node represents and to provide the information necessary for the method to be successfully executed.

4. Complete the following information in the dialog box:

   a. On the General Settings tab, click **Add** and select the **XML** option, select the correct XSD for the incoming XML file, and type a variable name for the incoming XML file.

      In the GetCustomer example, the variable name assigned is *requestCim*.

The following figure shows the General Settings tab for the *GetCustomer* example.



Specify a method name and select one or more parameter types.

Method Name: clientRequest

Settings:

Validate

(●) XML ( ) NonXML ( ) Java

mss/eventData.xsd
mss/orderStateNotification.xsd
sample/CimCustomerSample.xsc
Cim_Customer
Global Types
sample/HttpHeader.xsd
system/DynamicProperties.xsd
system/envelope.xsd
system/EventGeneratorMetadat

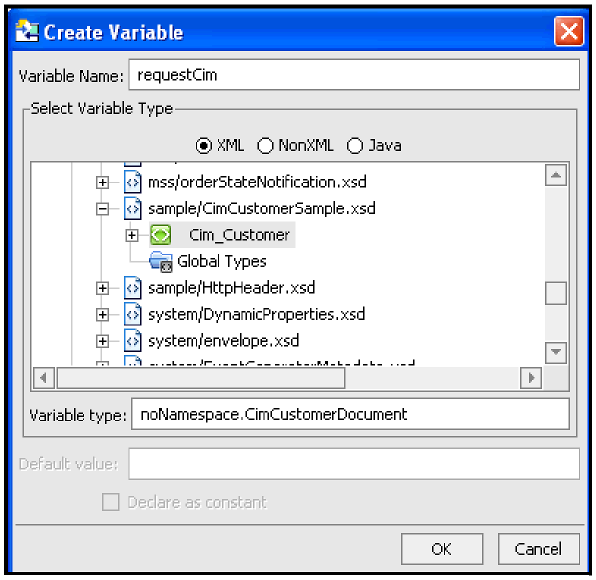
Type: space.CimCustomerDocument

Name: requestCim

b. Select the Receive Data tab, then in the Select Variables to Assign list, select Create New Variable.

The Create Variable dialog box appears. The Receive Data tab allows you to define variables that will be used for the incoming data.

c. Type the variable name, select the **XML** option, and select the appropriate XSD from the available list, just as you did on the General Settings tab, then click **OK**.



Variable Name: requestCim

Select Variable Type

(●) XML ( ) NonXML ( ) Java

mss/orderStateNotification.xsd
sample/CimCustomerSample.xsd
Cim_Customer
Global Types
sample/HttpHeader.xsd
system/DynamicProperties.xsd
system/envelope.xsd

Variable type: noNamespace.CimCustomerDocument

Default value:

Declare as constant

d. Close the Client Request dialog box.

Step 4. Adding a group to the workflow

The name for the group in the *GetCustomer* example is GetCustomerGroup. The group box allows you to pull nodes that have a process in common together. In this example, the group box holds nodes that share the same exception processing.

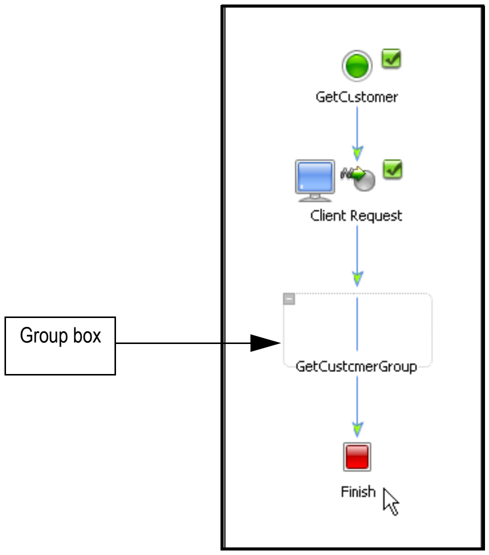
In this example, the group will contain three methods to accomplish the following tasks:

- Transform incoming data from the requestor's format into the MIM format understood by MetaSolv Solution
- Process the request to export customer information
- Transform data from the MIM format into the requestor's XML format

To add a group to the workflow

- From the Palette on the left side of the Workshop canvas, drag Group from the list and drop it on the workflow below the Client Request node.

  Type the name of the group on the canvas in the appropriate text box.



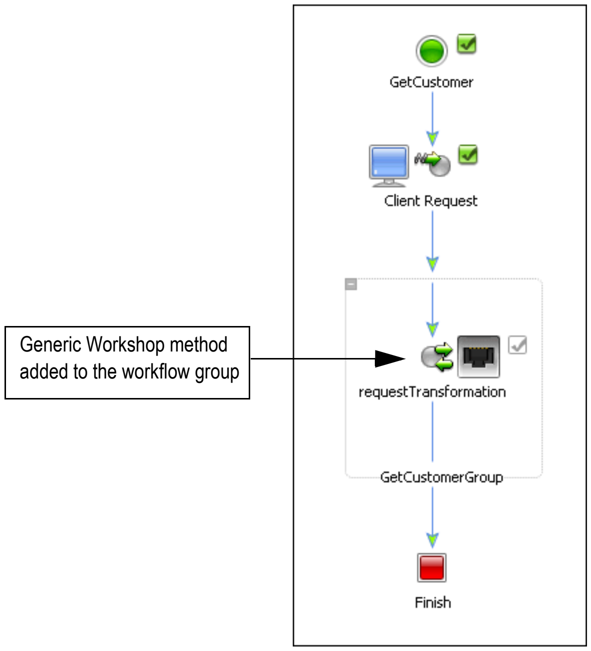
Step 5. Adding the request transformation method

This step adds the method to transform the data from the requestor's XML format into the MIM format used by MetaSolv Solution

To add the request transformation method

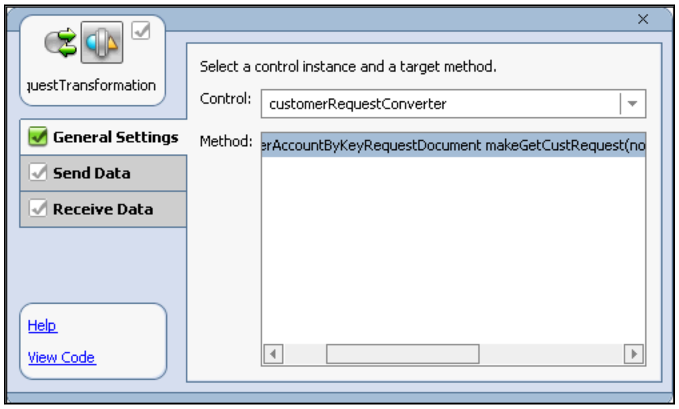
1. From the Palette on the left side of the Workshop canvas, drag **Control Send with Return** from the list, drop it into the group box, and type a name in the text box on the canvas.

   For the *GetCustomer* example, the name is *requestTransformation*. This is a generic Workshop method. You can use the MetaSolv Solution methods under the *MetaSolvCustomerManagement* control, but this demonstrates the use of a generic method.



2. Double-click the **requestTransformation** node.

The requestTransformation dialog box appears.



Select a control instance and a target method.

Control: customerRequestConverter

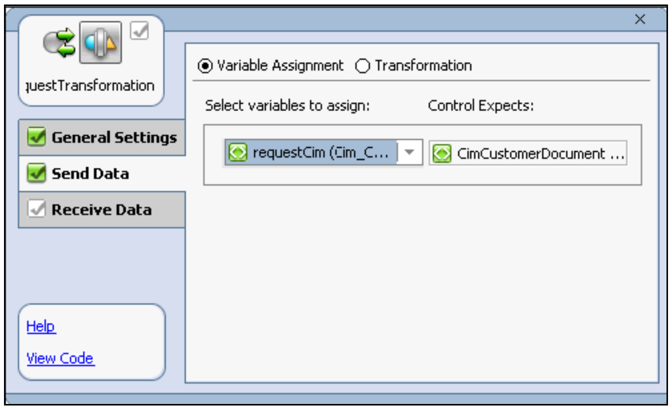
Method: erAccountByKeyRequestDocument makeGetCustRequest(no

3. On the General Settings tab, select the control and method to use for the request transformation.

   The control (*customerRequestConverter*) and its method (*makeGetCustRequest*) are selected. Because only one method was created, it is the only one available in the list box.

4. On the Send Data tab, select the variable that is to be assigned in this method for sending data to MetaSolv Solution.

   In this case, the send variable will be the variable identified in the clientRequest node, *requestCIM*.

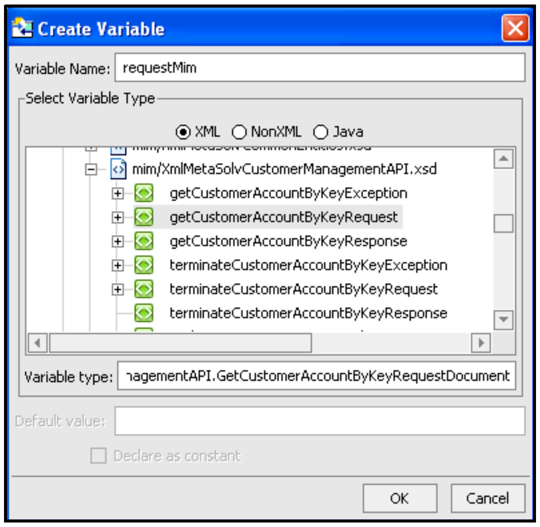


(● Variable Assignment) ○ Transformation

Select variables to assign:	Control Expects:
requestCim (Cim_C...	CimCustomerDocument ...

5. On the Receive Data tab, in the Variable to Assign list, select **Create New Variable**.

The following dialog box appears.The purpose of this step is to create a new variable to receive the transformed data.


Create Variable

Variable Name: requestMim

Select Variable Type

(● XML) ○ NonXML ○ Java

mim/XmlMetaSolvCustomerManagementAPI.xsd
- getCustomerAccountByKeyException
- getCustomerAccountByKeyRequest
- getCustomerAccountByKeyResponse
- terminateCustomerAccountByKeyException
- terminateCustomerAccountByKeyRequest
- terminateCustomerAccountByKeyResponse

Variable type: agementAPI.GetCustomerAccountByKeyRequestDocument

Default value:

☐ Declare as constant

6. Type a new name (*requestMim*) in the **Variable Name** field, select the **XML** option, select the appropriate schema from the list of available XSDs, and click **OK**.
7. Close the requestTransformation dialog box.

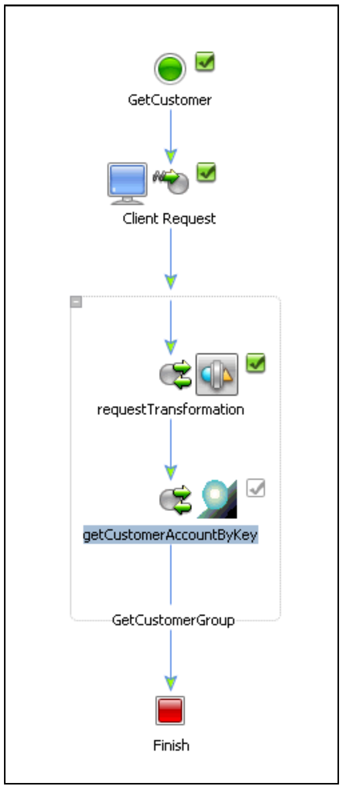
Step 6. Adding the method to process the request

Instead of a generic Workshop method that would have to be modified, this step uses a MetaSolv Customer Management API method created for processing this type of request.

To add the getCustomerAccountByKey method to the workflow

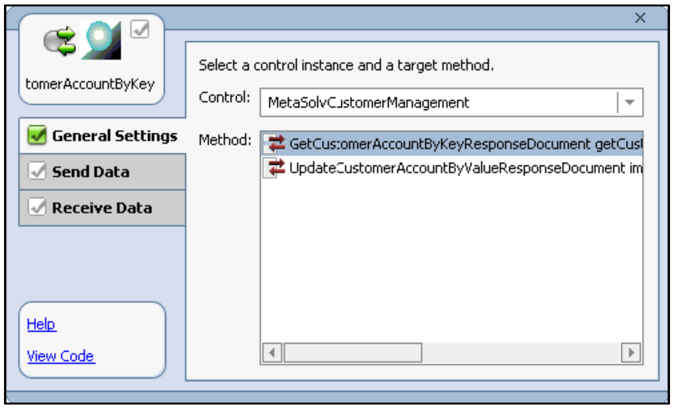
1. In the Control section of the Data Palette on the right side of the Workshop canvas, expand the MetaSolvCustomerManagement control to display its methods.
2. Locate and drag the getCustomerAccountByKey method to the workflow group and drop it below the requestTransformation node.

   The workflow now looks like the following figure.



3. Double-click the **getCustomerAccountByKeyRequest** node.

The getCustomerAccountByKey dialog box appears.

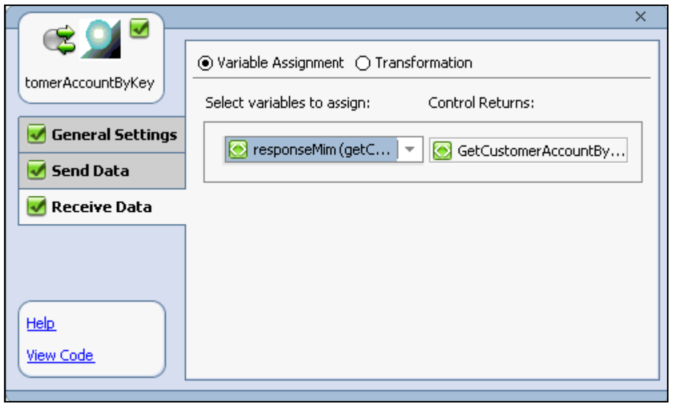


Select a control instance and a target method.

Control: MetaSolvCustomerManagement

Method: GetCustomerAccountByKeyResponseDocument getCust
UpdateCustomerAccountByValueResponseDocument im

4. On the General Settings tab, accept the defaults.
5. On the Send Data tab, in the Select Variables to Assign list box, select **requestMim**.
6. On the Receive Data tab, in the Select Variables to Assign list box, click the drop-down and select **Create New Variable,** then complete the following tasks in the window that appears:
   a. Type a name (*responseMim*) in the **Variable Name** field.
   b. Accept the variable type default.
   c. Click **OK**.



(●) Variable Assignment ( ) Transformation

Select variables to assign: responseMim (getC...

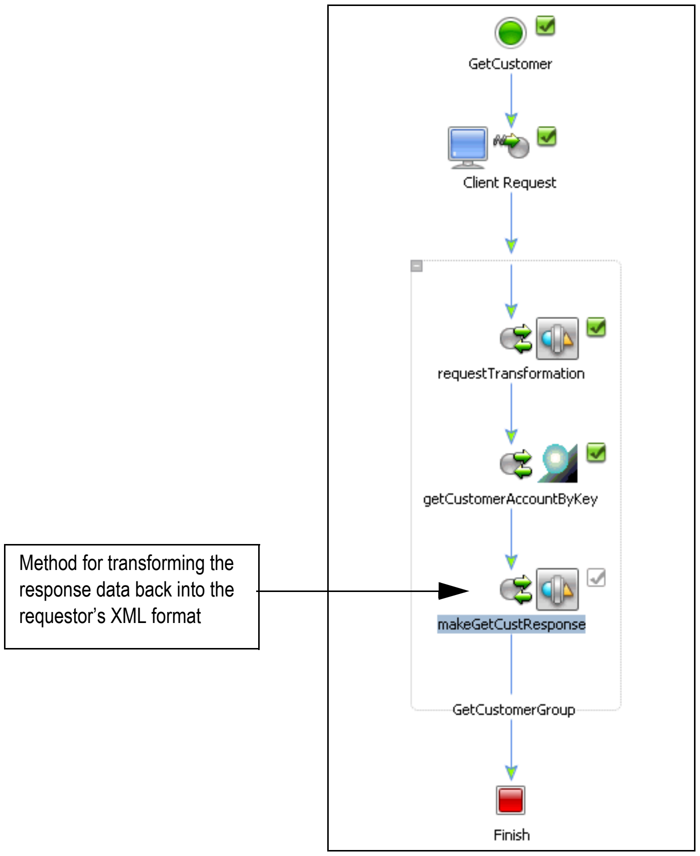
Control Returns: GetCustomerAccountBy...

7. Close the getCustomerAccountByKey dialog box.

Step 7. Adding the response transformation method

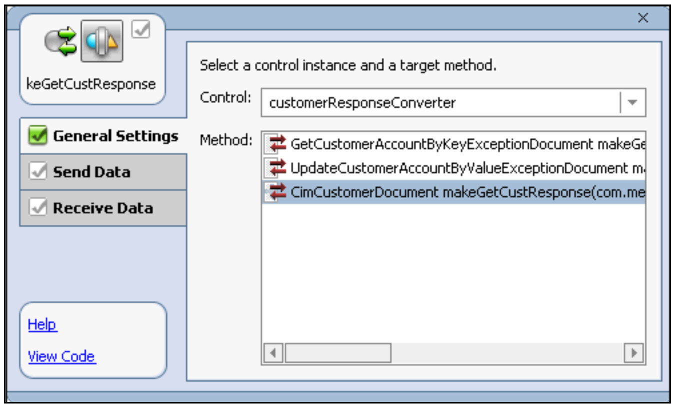
To add the response transformation method

1. In the Controls section of the Data Palette, expand the *customerResponseConverter* control, then drag the **makeGetCustResponse** method to the group on the Workshop canvas and drop it under the getCustomerAccountByKey node.



2. Double-click the **makeGetCustResponse** node.

The makeGetCustResponse dialog box appears.

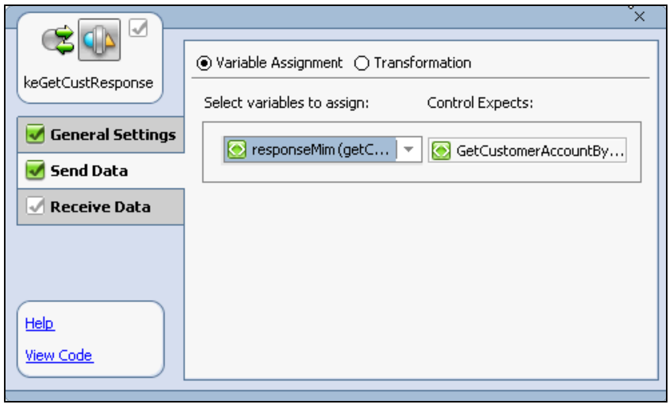


Select a control instance and a target method.

Control: customerResponseConverter

Method:
GetCustomerAccountByKeyExceptionDocument makeGe
UpdateCustomerAccountByValueExceptionDocument m
CimCustomerDocument makeGetCustResponse(com.me

3. On the General Settings tab, accept the defaults.
4. On the Send Data tab, in the **Select variables to assign** list, select **responseMim.**



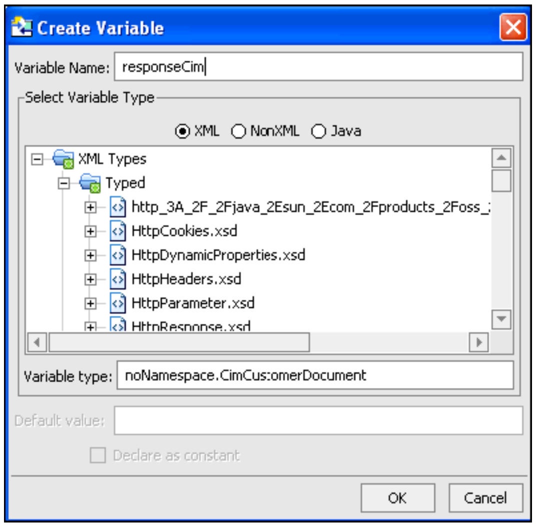
(● Variable Assignment) ○ Transformation

Select variables to assign:	Control Expects:
responseMim (getC…	GetCustomerAccountBy…

5. On the Receive Data tab, in the **Select variables to assign** list, select **Create New Variable**.

   The Create Variable dialog box appears.
6. In the **Variable Name** field, type **responseCim.**
7. Accept the default variable type and click OK.

   The following figure shows the Create Variable dialog box with the values entered.



Variable Name: responseCim

[✓ XML] NonXML Java

http_3A_2F_2Fjava_2Esun_2Ecom_2Fproducts_2Foss_
HttpCookies.xsd
HttpDynamicProperties.xsd
HttpHeaders.xsd
HttpParameter.xsd
HttpResponse.xsd

Variable type: noNamespace.CimCustomerDocument

Default value:

Declare as constant

8. Close the makeGetCustResponse dialog box.
9. In the palette on the left pane of Workshop, select Client Response in the list, drag it to the canvas, and drop it in the group under the makeGetCustResponse node.
10. Double-click the Client Response node.

    The Client Response dialog box appears.
11. On the General Settings tab, click **Add,** and in the window that appears complete the following tasks:

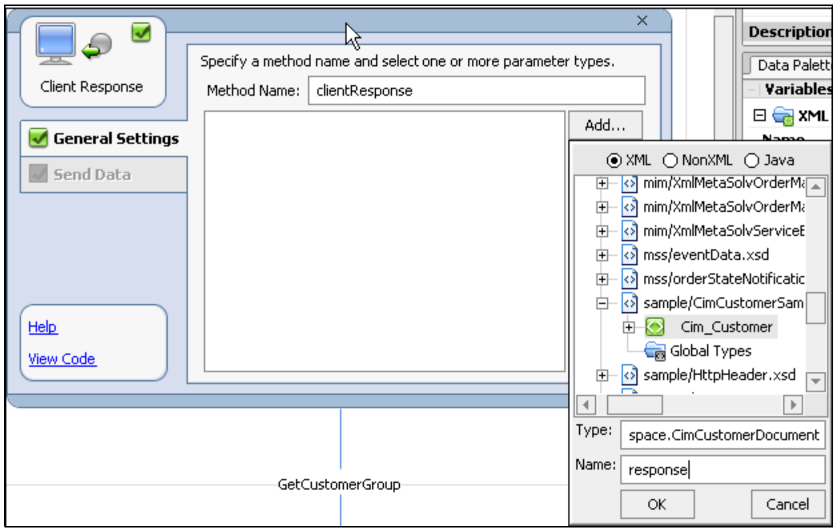
    a. Expand the CimCustomerSample.xsd and select **Cim_Customer**

    b. Type a name in the **Name** field.

       In the case of the GetCustomer example, the name for the is *response*.

    c. Click **OK**.

       The following figure shows the General Settings tab.



Specify a method name and select one or more parameter types.

Method Name: clientResponse

(•) XML ( ) NonXML ( ) Java

- mim/XmlMetaSolvOrderM
- mim/XmlMetaSolvOrderM
- mim/XmlMetaSolvServiceE
- mss/eventData.xsd
- mss/orderStateNotificatic
- sample/CimCustomerSam
  - Cim_Customer
  - Global Types
- sample/HttpHeader.xsd

Type: space.CimCustomerDocument

Name: response

12. On the Send Data tab, select the **responseCIM** variable and close the Client Response dialog box.

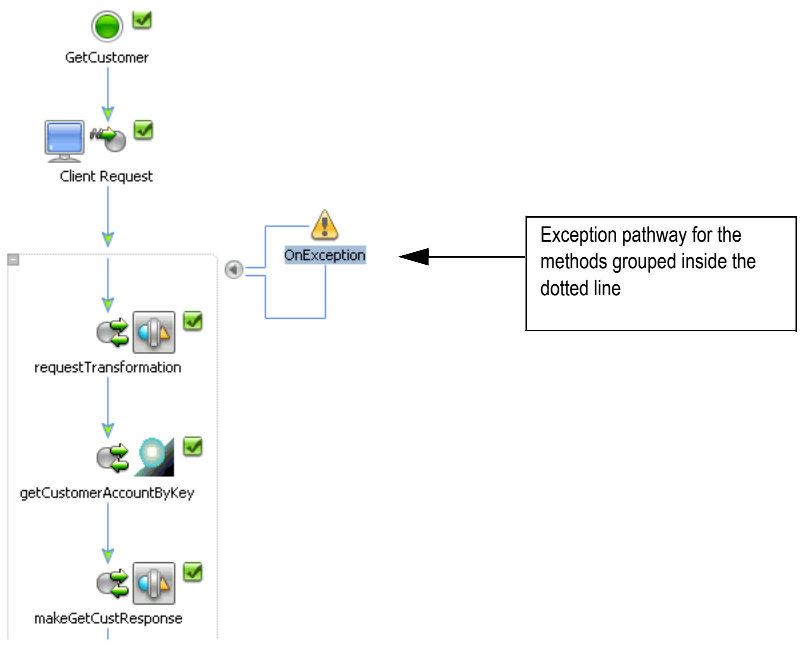
Step 8. Setting up exception handling

This section explains how to handle exceptions in the workflow.

To set up exception handling for the workflow

1. Place the cursor in the group to which exception handling is to apply, right-click, and select **Add exception path.**

   The exception path is added to the group that you indicate.

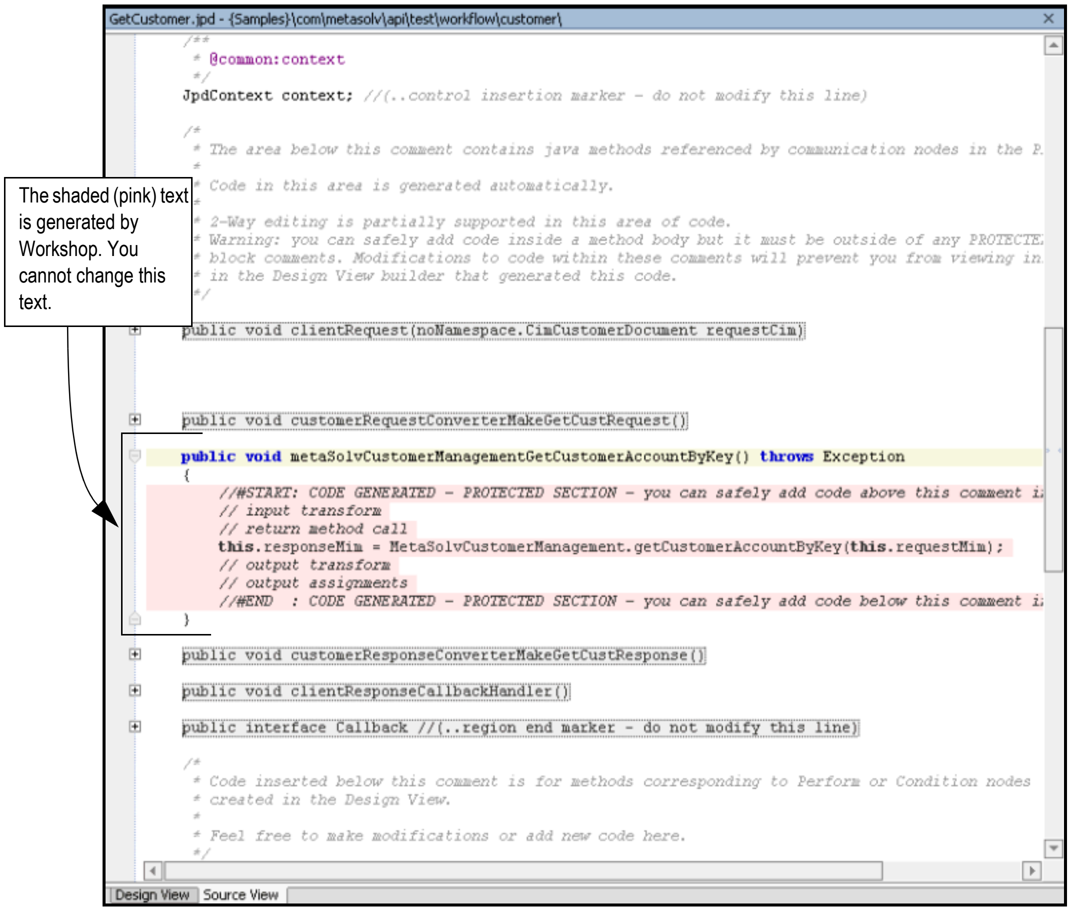


   To catch any exception returned by the XML APIs for one of the methods in the group, you must set up a **try-catch** expression. This requires that you go into text view and enclose the appropriate code with the expression.

   Error data is logged in a file called appserverlog.xml

2. Right-click on the getCustomerAccountbyKey node and select **View code**.

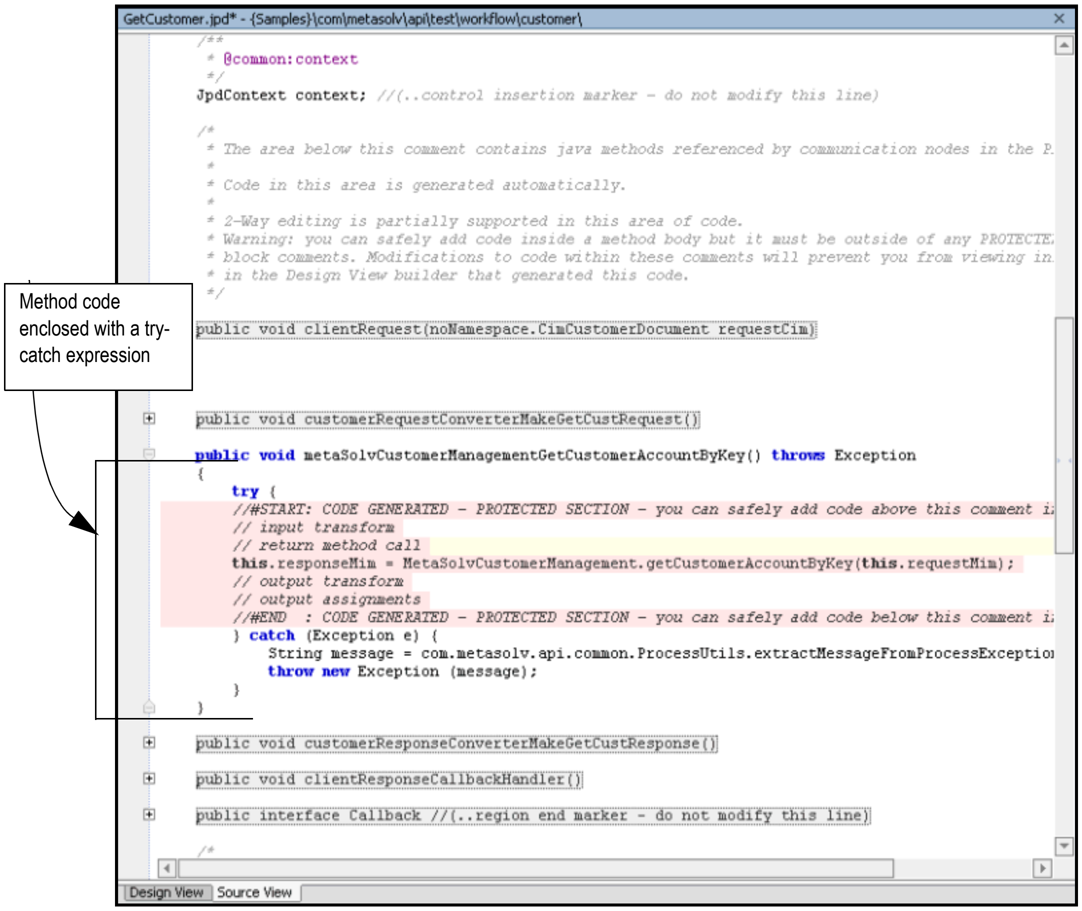
The code is displayed in text. See the following figure.


GetCustomer.jpd - {Samples}\com\metasolv\api\test\workflow\customer\
The shaded (pink) text is generated by Workshop. You cannot change this text.
Design View
Source View

```
/**
 * @common:context
 */
JpdContext context; //(..control insertion marker - do not modify this line)

/*
 * The area below this comment contains java methods referenced by communication nodes in the P
 *
 * Code in this area is generated automatically.
 *
 * 2-Way editing is partially supported in this area of code.
 * Warning: you can safely add code inside a method body but it must be outside of any PROTECTE
 * block comments. Modifications to code within these comments will prevent you from viewing in
 * in the Design View builder that generated this code.
 */
public void clientRequest(noNamespace.CimCustomerDocument requestCim)



public void customerRequestConverterMakeGetCustRequest()

public void metaSolvCustomerManagementGetCustomerAccountByKey() throws Exception
{
    //#START: CODE GENERATED - PROTECTED SECTION - you can safely add code above this comment i
    // input transform
    // return method call
    this.responseMim = MetaSolvCustomerManagement.getCustomerAccountByKey(this.requestMim);
    // output transform
    // output assignments
    //#END  : CODE GENERATED - PROTECTED SECTION - you can safely add code below this comment i
}
public void customerResponseConverterMakeGetCustResponse()

public void clientResponseCallbackHandler()

public interface Callback //(..region end marker - do not modify this line)

/*
 * Code inserted below this comment is for methods corresponding to Perform or Condition nodes
 * created in the Design View.
 *
 * Feel free to make modifications or add new code here.
 */
```

3. Enclose the code in a try-catch expression.

The following figure shows the code with a try-catch expression enclosing the code.

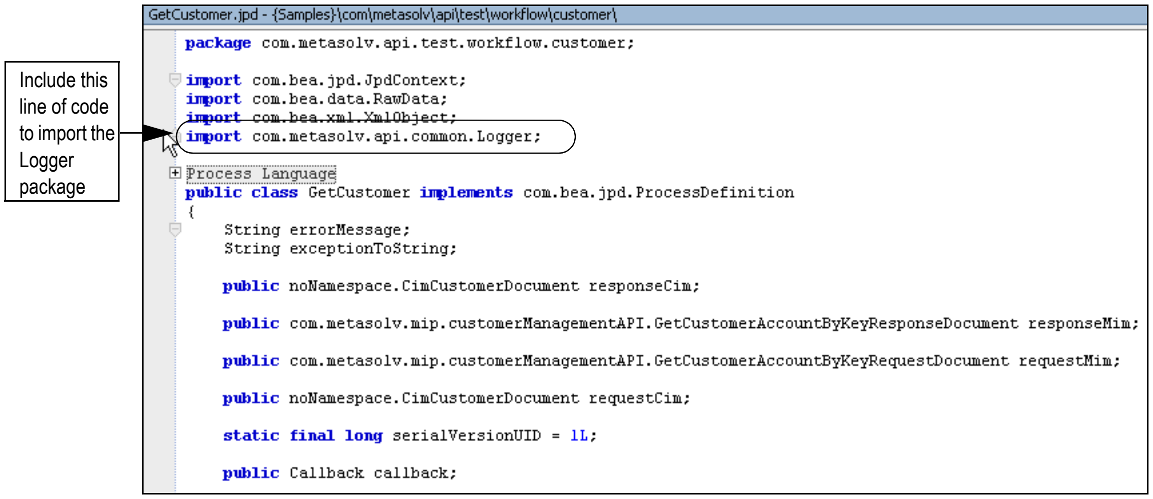

GetCustomer.jpd* - {Samples}\com\metasolv\api\test\workflow\customer\
Method code enclosed with a try-catch expression
Design View
Source View

```
/**
 * @common:context
 */
JpdContext context; //(..control insertion marker - do not modify this line)

/*
 * The area below this comment contains java methods referenced by communication nodes in the P
 *
 * Code in this area is generated automatically.
 *
 * 2-Way editing is partially supported in this area of code.
 * Warning: you can safely add code inside a method body but it must be outside of any PROTECTE
 * block comments. Modifications to code within these comments will prevent you from viewing in
 * in the Design View builder that generated this code.
 */

public void clientRequest(noNamespace.CimCustomerDocument requestCim)


public void customerRequestConverterMakeGetCustRequest()

public void metaSolvCustomerManagementGetCustomerAccountByKey() throws Exception
{
    try {
    //#START: CODE GENERATED - PROTECTED SECTION - you can safely add code above this comment i
    // input transform
    // return method call
    this.responseMim = MetaSolvCustomerManagement.getCustomerAccountByKey(this.requestMim);
    // output transform
    // output assignments
    //#END  : CODE GENERATED - PROTECTED SECTION - you can safely add code below this comment i
    } catch (Exception e) {
        String message = com.metasolv.api.common.ProcessUtils.extractMessageFromProcessExceptio
        throw new Exception (message);
    }
}

public void customerResponseConverterMakeGetCustResponse()

public void clientResponseCallbackHandler()

public interface Callback //(..region end marker - do not modify this line)

/*
```

The try-catch expression is for external processes that occur in the MetaSolv Solution core. The transformation processes are local and do not require a try-catch expression.

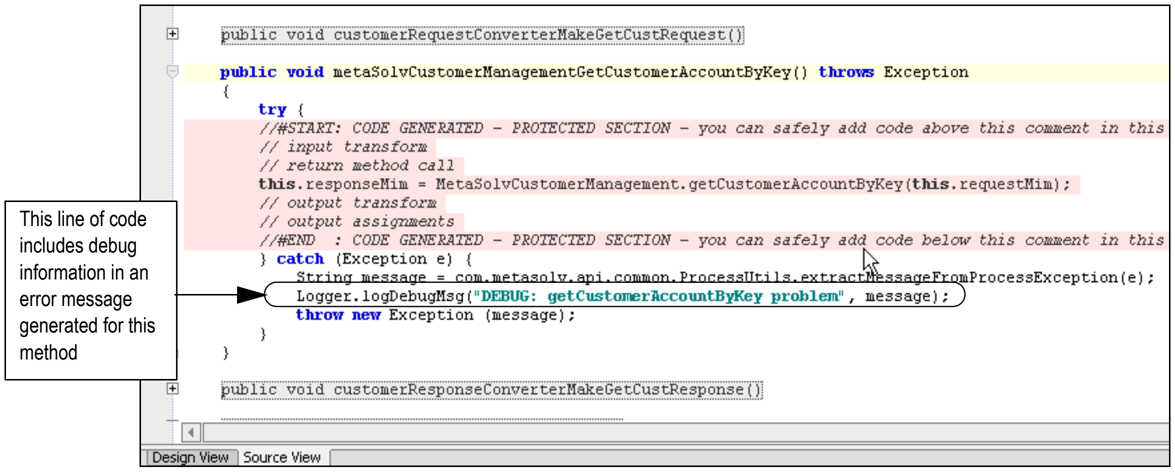
4. Include code for logging.

   The following figures show how to add code for logging.



GetCustomer.jpd - {Samples}\com\metasolv\api\test\workflow\customer\

```
package com.metasolv.api.test.workflow.customer;

import com.bea.jpd.JpdContext;
import com.bea.data.RawData;
import com.bea.xml.XmlObject;
import com.metasolv.api.common.Logger;

Process Language
public class GetCustomer implements com.bea.jpd.ProcessDefinition
{
    String errorMessage;
    String exceptionToString;

    public noNamespace.CimCustomerDocument responseCim;

    public com.metasolv.mip.customerManagementAPI.GetCustomerAccountByKeyResponseDocument responseMim;

    public com.metasolv.mip.customerManagementAPI.GetCustomerAccountByKeyRequestDocument requestMim;

    public noNamespace.CimCustomerDocument requestCim;

    static final long serialVersionUID = 1L;

    public Callback callback;
```


This line of code includes debug information in an error message generated for this method
Design View
Source View

```
    public void customerRequestConverterMakeGetCustRequest()

    public void metaSolvCustomerManagementGetCustomerAccountByKey() throws Exception
    {
        try {
        //#START: CODE GENERATED - PROTECTED SECTION - you can safely add code above this comment in this
        // input transform
        // return method call
        this.responseMim = MetaSolvCustomerManagement.getCustomerAccountByKey(this.requestMim);
        // output transform
        // output assignments
        //#END  : CODE GENERATED - PROTECTED SECTION - you can safely add code below this comment in this
        } catch (Exception e) {
            String message = com.metasolv.api.common.ProcessUtils.extractMessageFromProcessException(e);
            Logger.logDebugMsg("DEBUG: getCustomerAccountByKey problem", message);
            throw new Exception (message);
        }
    }

    public void customerResponseConverterMakeGetCustResponse()
```

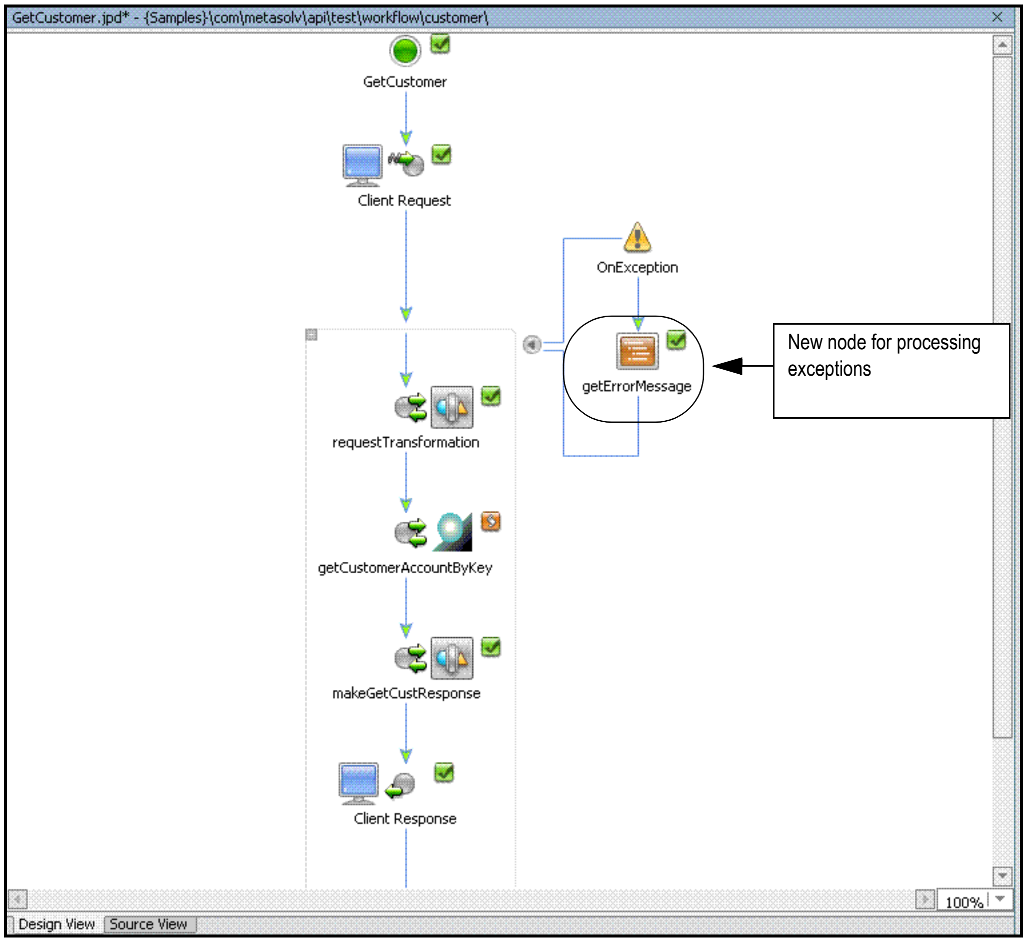
5. Click the Design View tab at the bottom of the Workshop canvas to return to the graphical view of the workflow.

6. Complete the following actions to capture exceptions and assign them to variables in the workflow.

   a. In the Palette on the left pane, locate the Perform node, then drag it to the exception path in the workflow and drop it on the line below the Exception node.

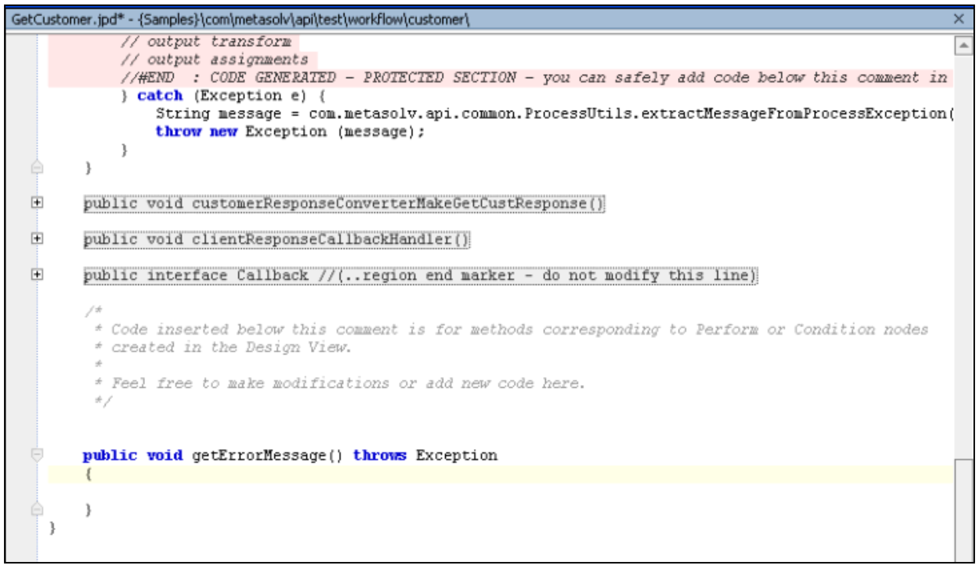
   b. Type a name for the new node.

For example, *getErrorMessage*. The following figure shows the new node in the workflow.



7. Double-click the getErrorMessage node.
8. In the dialog box that appears, type getErrorMessage in the **Name** field and click the **View Code** link.

The text view appears.



GetCustomer.jpd* - {Samples}\com\metasolv\api\test\workflow\customer\

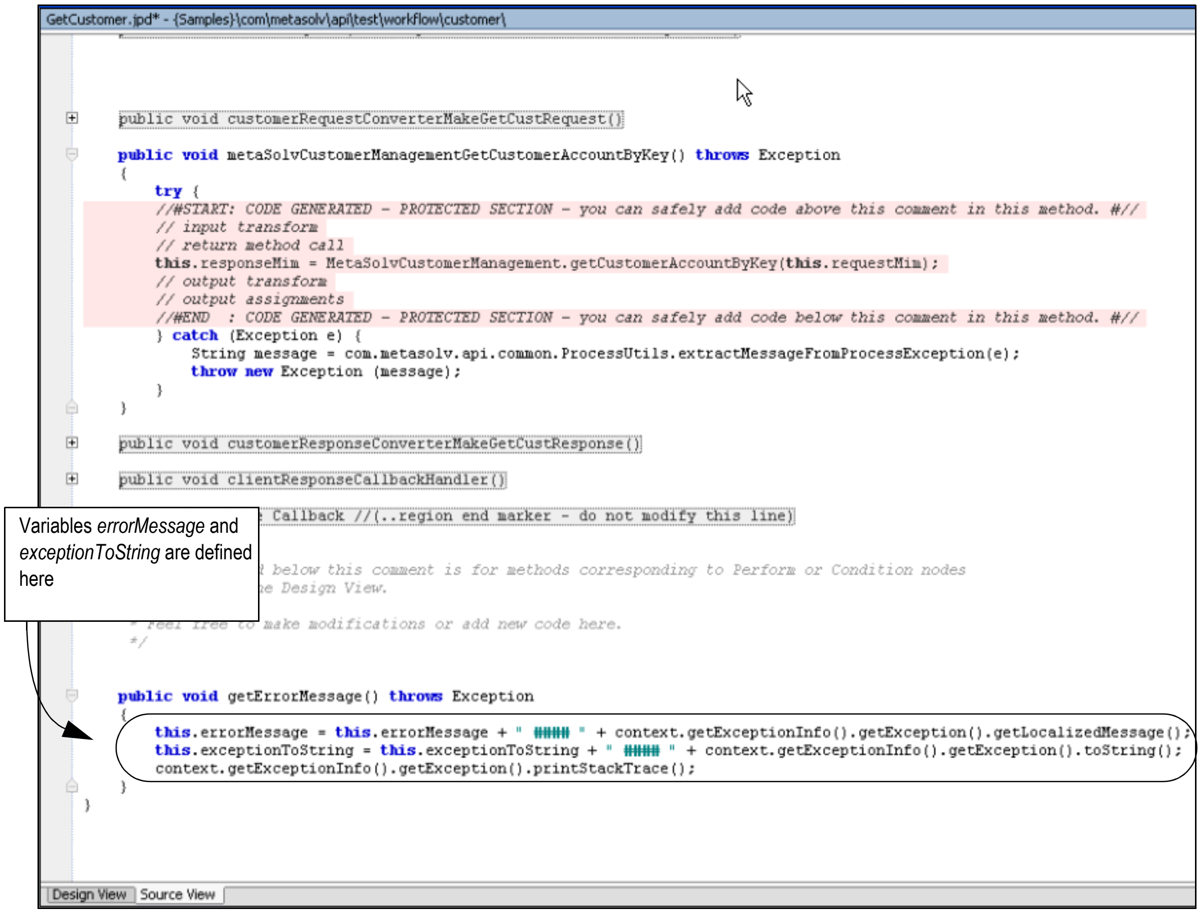
```
            // output transform
            // output assignments
            //#END  : CODE GENERATED - PROTECTED SECTION - you can safely add code below this comment in
            } catch (Exception e) {
                String message = com.metasolv.api.common.ProcessUtils.extractMessageFromProcessException(
                throw new Exception (message);
            }
        }

        public void customerResponseConverterMakeGetCustResponse()

        public void clientResponseCallbackHandler()

        public interface Callback //(..region end marker - do not modify this line)

        /*
         * Code inserted below this comment is for methods corresponding to Perform or Condition nodes
         * created in the Design View.
         *
         * Feel free to make modifications or add new code here.
         */


        public void getErrorMessage() throws Exception
        {

        }
    }
```

9. Manually define the following variables in the source view:

   errorMessage

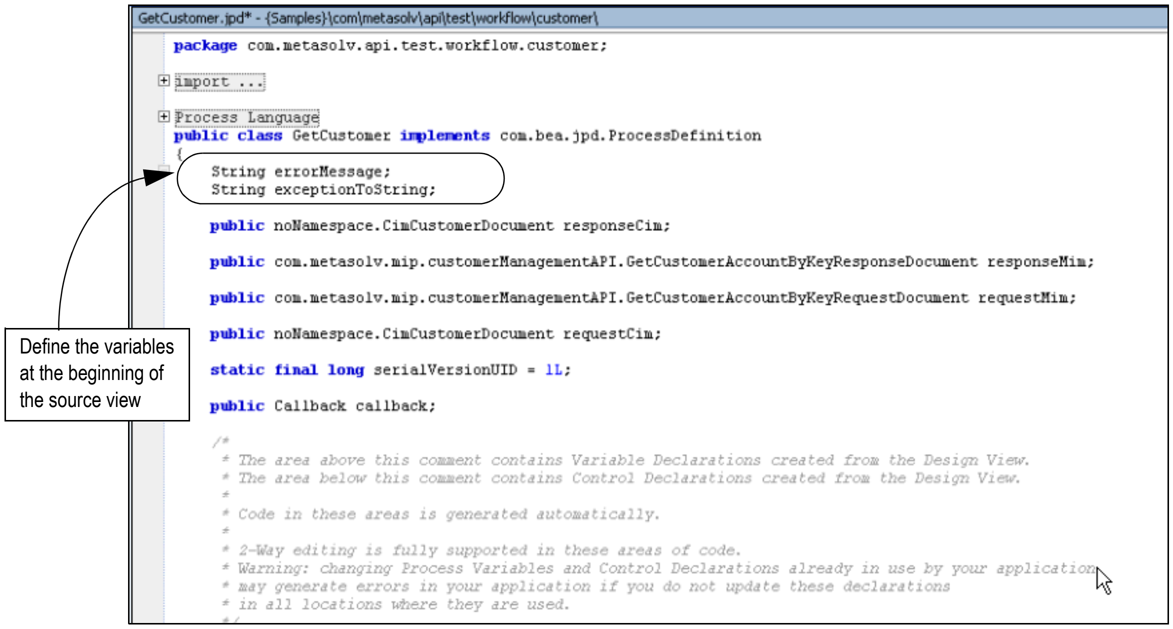
   exceptionToString

   The following figure shows the variables entered in the source view.


GetCustomer.jpd* - {Samples}\com\metasolv\api\test\workflow\customer\
Variables *errorMessage* and *exceptionToString* are defined here
Design View
Source View

```
    public void customerRequestConverterMakeGetCustRequest()

    public void metaSolvCustomerManagementGetCustomerAccountByKey() throws Exception
    {
        try {
        //#START: CODE GENERATED - PROTECTED SECTION - you can safely add code above this comment in this method. #//
        // input transform
        // return method call
        this.responseMim = MetaSolvCustomerManagement.getCustomerAccountByKey(this.requestMim);
        // output transform
        // output assignments
        //#END  : CODE GENERATED - PROTECTED SECTION - you can safely add code below this comment in this method. #//
        } catch (Exception e) {
            String message = com.metasolv.api.common.ProcessUtils.extractMessageFromProcessException(e);
            throw new Exception (message);
        }
    }

    public void customerResponseConverterMakeGetCustResponse()

    public void clientResponseCallbackHandler()

            Callback //(..region end marker - do not modify this line)

            below this comment is for methods corresponding to Perform or Condition nodes
            e Design View.

     * Feel free to make modifications or add new code here.
     */


    public void getErrorMessage() throws Exception
    {
        this.errorMessage = this.errorMessage + " ##### " + context.getExceptionInfo().getException().getLocalizedMessage();
        this.exceptionToString = this.exceptionToString + " ##### " + context.getExceptionInfo().getException().toString();
        context.getExceptionInfo().getException().printStackTrace();
    }
}
```

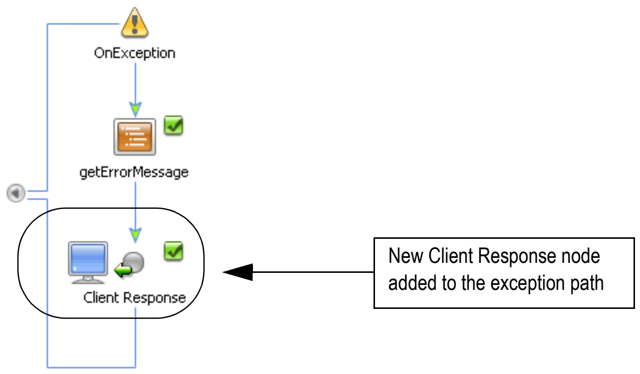
10. Complete the definition of the variables by scrolling to the beginning of the source view and making the changes to the code shown in the following figure.


GetCustomer.jpd* - {Samples}\com\metasolv\api\test\workflow\customer\
Define the variables at the beginning of the source view

```
package com.metasolv.api.test.workflow.customer;

import ...

Process Language
public class GetCustomer implements com.bea.jpd.ProcessDefinition
{
    String errorMessage;
    String exceptionToString;

    public noNamespace.CimCustomerDocument responseCim;

    public com.metasolv.mip.customerManagementAPI.GetCustomerAccountByKeyResponseDocument responseMim;

    public com.metasolv.mip.customerManagementAPI.GetCustomerAccountByKeyRequestDocument requestMim;

    public noNamespace.CimCustomerDocument requestCim;

    static final long serialVersionUID = 1L;

    public Callback callback;

    /*
     * The area above this comment contains Variable Declarations created from the Design View.
     * The area below this comment contains Control Declarations created from the Design View.
     *
     * Code in these areas is generated automatically.
     *
     * 2-Way editing is fully supported in these areas of code.
     * Warning: changing Process Variables and Control Declarations already in use by your application
     * may generate errors in your application if you do not update these declarations
     * in all locations where they are used.
     */
```

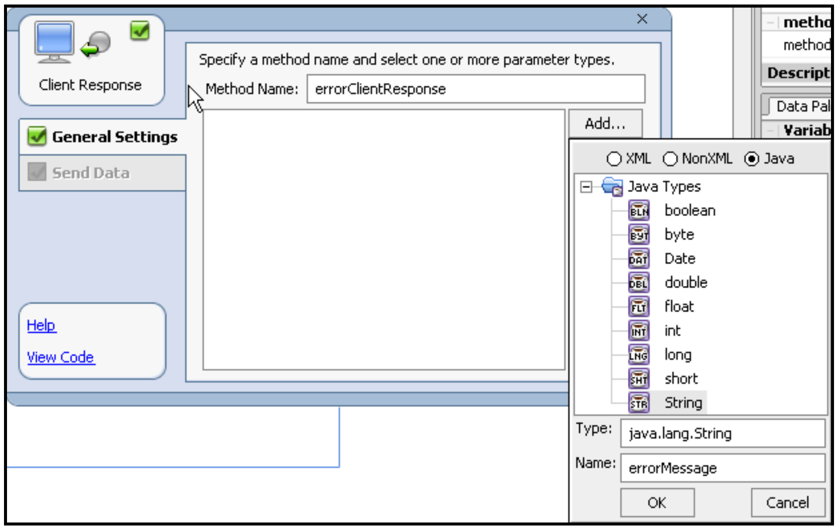
11. Click Design View at the bottom of the canvas and close the getErrorMessage dialog box.
12. On the left palette, locate the Client Response method and drag it to the exception path and drop it below the getErrorMessage node.

    The following figure shows the Client Response node in the exception path.



13. Double-click the Client Response node.

The Client Response dialog box appears.



Specify a method name and select one or more parameter types.

Method Name: errorClientResponse

XML [✓ Java] NonXML

Type: java.lang.String

Name: errorMessage

14. On the General Settings tab, complete the following tasks:

    a. Type a method name.

       For example *errorClientResponse*.

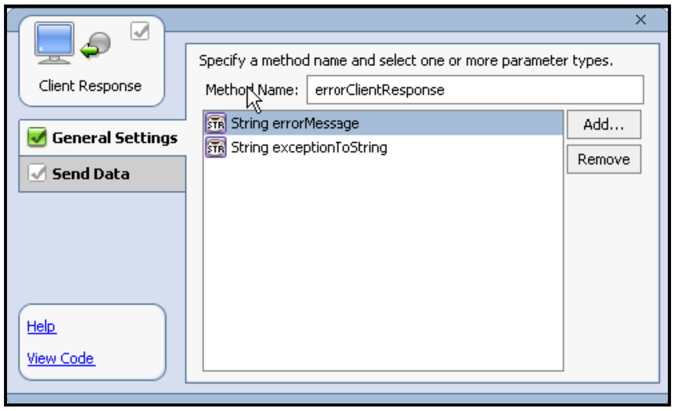
    b. Click **Add**.

    c. In the window that appears, select the **Java** option, **String** from the Java types list, and type a the name of one of the exception variables that was defined manually in the **Name** field (for example, *errorMessage*).

    d. Click **OK**.

    e. Repeat steps b-d to add the other exception variable that was defined manually (exceptionToString).

The following figure shows both variables defined for the Client Response method.



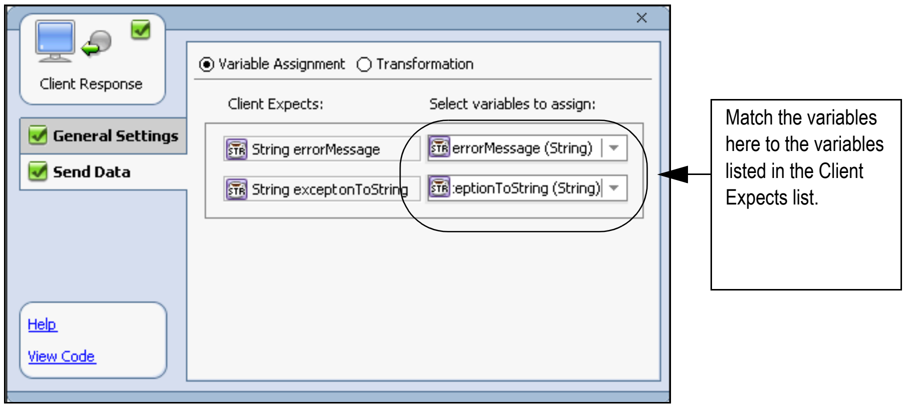
Specify a method name and select one or more parameter types.

Method Name: errorClientResponse

String errorMessage
String exceptionToString

15. On the Send Data tab, for each variable in the Client Expects list, select the appropriate variable in the **Select variables to assign** list.

    See the following figure for more information.

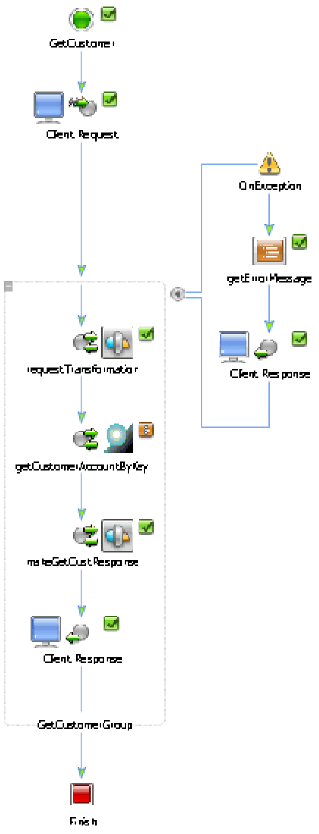

Client Response
General Settings
Send Data
Help
View Code
Match the variables here to the variables listed in the Client Expects list.

[✓ Variable Assignment] Transformation

Client Expects:	Select variables to assign:
String errorMessage	errorMessage (String)
String exceptonToString	:eptionToString (String)

16. Close the Client Response dialog box.

The following figure shows the completed workflow.



17. Click **Save** on the main menu to save the workflow.

Testing the application in Workshop

When the workflow is complete, you can test it in Workshop.

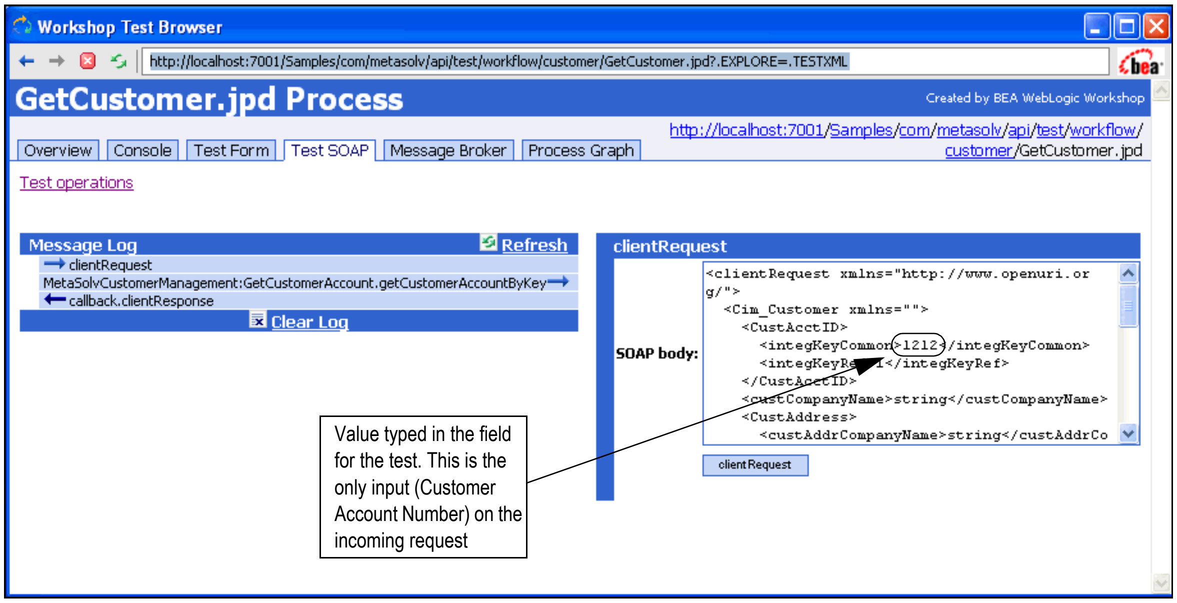
To test a workflow

1. On the Workshop main menu, select **Debug>Start**.

   The Workshop test browser opens.

2. Select the Test SOAP tab.

   Workshop creates an XML file for the Client Request method based on the requestor's XSD. In the case of the example used in this chapter, the test XML file was created from the Cim_Customer XSD.


Workshop Test Browser
http://localhost:7001/Samples/com/metasolv/api/test/workflow/customer/GetCustomer.jpd?.EXPLORE=.TESTXML
GetCustomer.jpd Process
Created by BEA WebLogic Workshop
http://localhost:7001/Samples/com/metasolv/api/test/workflow/customer/GetCustomer.jpd
Overview
Console
Test Form
Test SOAP
Message Broker
Process Graph
Test operations
Message Log
Refresh
Clear Log
clientRequest
SOAP body:
clientRequest
Value typed in the field for the test. This is the only input (Customer Account Number) on the incoming request

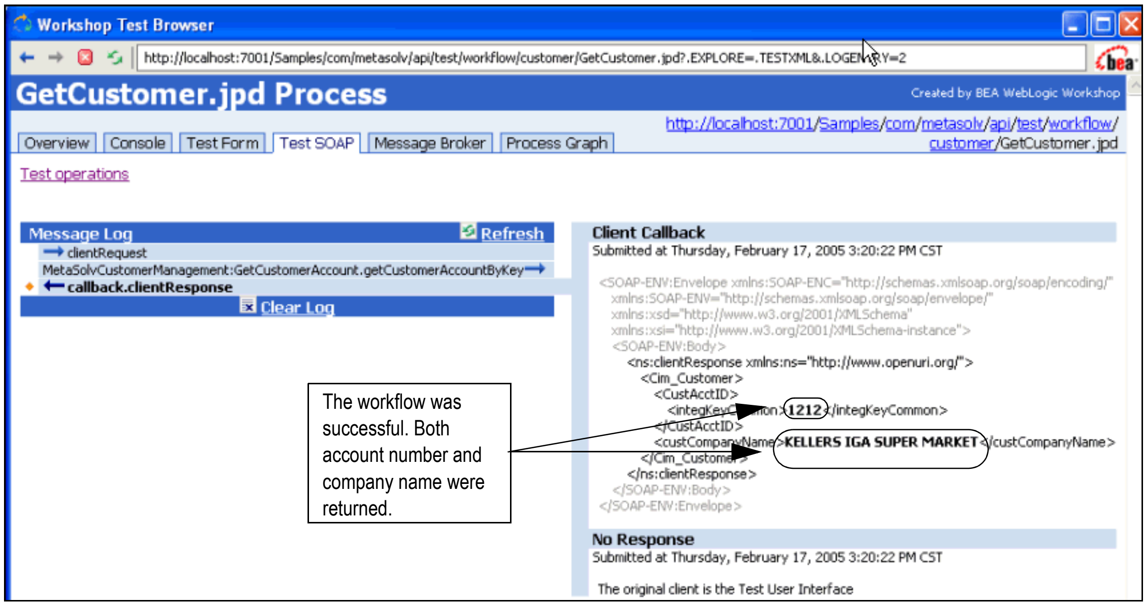
```
clientRequest
MetaSolvCustomerManagement:GetCustomerAccount.getCustomerAccountByKey
callback.clientResponse
```

```
<clientRequest xmlns="http://www.openuri.or
g/">
  <Cim_Customer xmlns="">
    <CustAcctID>
      <integKeyCommon>1212</integKeyCommon>
      <integKeyRef>1</integKeyRef>
    </CustAcctID>
    <custCompanyName>string</custCompanyName>
    <CustAddress>
      <custAddrCompanyName>string</custAddrCo
```

3. Type in your test data in the appropriate fields in the test XML file, as shown in the previous figure.

4. Click **client Request**.

Workshop tests the workflow and indicates the results of the test. See the following figure.


Workshop Test Browser
http://localhost:7001/Samples/com/metasolv/api/test/workflow/customer/GetCustomer.jpd?.EXPLORE=.TESTXML&.LOGENTRY=2
GetCustomer.jpd Process
Created by BEA WebLogic Workshop
http://localhost:7001/Samples/com/metasolv/api/test/workflow/customer/GetCustomer.jpd
Overview
Console
Test Form
Test SOAP
Message Broker
Process Graph
Test operations
Message Log
Refresh
clientRequest
MetaSolvCustomerManagement:GetCustomerAccount.getCustomerAccountByKey
callback.clientResponse
Clear Log

The workflow was successful. Both account number and company name were returned.

Client Callback
Submitted at Thursday, February 17, 2005 3:20:22 PM CST

```
<SOAP-ENV:Envelope xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <SOAP-ENV:Body>
    <ns:clientResponse xmlns:ns="http://www.openuri.org/">
      <Cim_Customer>
        <CustAcctID>
          <integKeyCommon>1212</integKeyCommon>
        </CustAcctID>
        <custCompanyName>KELLERS IGA SUPER MARKET</custCompanyName>
      </Cim_Customer>
    </ns:clientResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

No Response
Submitted at Thursday, February 17, 2005 3:20:22 PM CST

The original client is the Test User Interface

5. Click **callback.clientResponse** in the Message Log listing to see the XML that was returned as a response.

Creating a build

To create a build, select **Build>Build Application** from the main menu.

To create a build and an ear file, select **Build>Build ear**. When you create an ear file in Workshop, it is automatically deployed to WebLogic Server.

4

Post Development Tasks

Updating the production database

During development, the MetaSolv Solution XML API application requires the creation of a number of tables within the database used for WebLogic Integration conversation state tracking. The related database is defined within the WebLogic data-source named bpmArchDataSource. This data-source is configured during creation of the WebLogic domain using the Configuration wizard. To move the application into a production mode, you must recreate the tables in the production database.

To create the production database tables, you must complete the following tasks:

1. Create an SQL script to create tables in the production Oracle database.
2. Run the SQL script against the production database to add the tables.

Creating the SQL script

The following shell scripts *tableTool.sh* can be uses to generate the Integration state tables:

Windows: tableTool.bat

UNIX: tableTool.sh

Prerequisites

You must have BEA WebLogic server.

To create the SQL script

1. Copy the following files to a directory on the server:
   - ManifestTableGenerationTemplate.xsl
   - ManifestTableRemovalTemplate.xsl
   - tableTool.class
   - tableTool.sh
2. To generate the Integration State Tables SQL scripts, execute the tableTool.sh script with the following arguments:

   ./tableTool.sh [*BEA_JAVA_HOME*] [*MIP.ear*] [*createTable* | *dropTable*]

Arguments:

[*BEA_JAVA_HOME*] is the location of the BEA JDK.
For example: /opt/bea/jdk142_04

[*MIP.ear*] is the location of the MIP.ear.
For example: /opt/bea/user_projects/domains/paetec/lib/MIP.ear

[*createTable* | *dropTable*] If **createTable** is specified, the SQL script tableGenerationTemplate.sql is generated.The SQL script can be used to create the Integration state tables.

If **dropTable** is specified, the SQL script tableRemovalTemplate.sql is generated. This SQL script can be used to drop the Integration state tables.

Example: The following example shows the command to generate a create table SQL script for the Integration state tables:

```
./tableTool.sh /opt/bea/jdk142_04 /opt/bea/user_projects/domains/newtel/lib/MIP.ear
createTable
```

Example The following example shows the command to generate the drop table SQL script for the Integration state tables:

```
./tableTool.sh /opt/bea/jdk142_04 /opt/bea/user_projects/domains/newtel/lib/MIP.ear
dropTable
```

Running the SQL script

The SQL file is designed for configuring an Oracle database on a UNIX or Windows platform. The SQL syntax may vary slightly by database vendor. Modifications to the syntax of the commands may be required for successful creation of the tables. Some tables may already be present.

If some commands do not execute due to pre-existence of the table, you may ignore the error. A number of the tables within the script are common tables required by integration applications and may already be present on your system.

Note that the SQL file to drop tables is provided for your convenience.

The Java Naming and Directory Interface (JNDI) name must be exactly as shown. The names are case sensitive. Names must include the periods (.) and underscores (_) as shown.

To run the SQL script

1. Connect to the WebLogic Integration database as a user having create table privileges.
2. Run the SQL file.

Setting up gateway events

In addition to the configuration required to make the MetaSolv Solution adapter functional at installation, MetaSolv Solution must be configured to receive data and return data.

The Integration server continuously checks the MetaSolv Solution database for events that are ready to be sent to an external application. The Integration server also monitors the external application for updates to the status of a gateway event. When an external application sends a status update, the Integration server records the new status in the MetaSolv Solution database.

The following sections describe how to set up gateway events for communication between MetaSolv Solution and the adapter.

Creating a gateway event

Gateway events are set up in the MetaSolv Solution user interface. Complete information on how to create a gateway event is located in MetaSolv Solution Help. This section explains:

- Basic steps for creating a gateway event
- How to find the exact procedures for creating a gateway event in Help.

Basic steps for creating a gateway event in the MetaSolv Solution interface

1. Create a new gateway event.

   For example *xxx_mip_order_event.* This is done on the Gateway window in MetaSolv Solution. MetaSolv Solution creates a gatewayEvent and assigns an eventID.

2. Add a binding to the gateway.

   Set the Binding Type to **IOR** and provide the location path to the NameService.ior file. The path should be similar to the following examples:

   Windows

   ```
   c:\Metasolv\Appserver\IOR\NameService.ior
   ```

   UNIX

   ```
   Metasolv/Appserver/IOR/NameService.ior
   ```

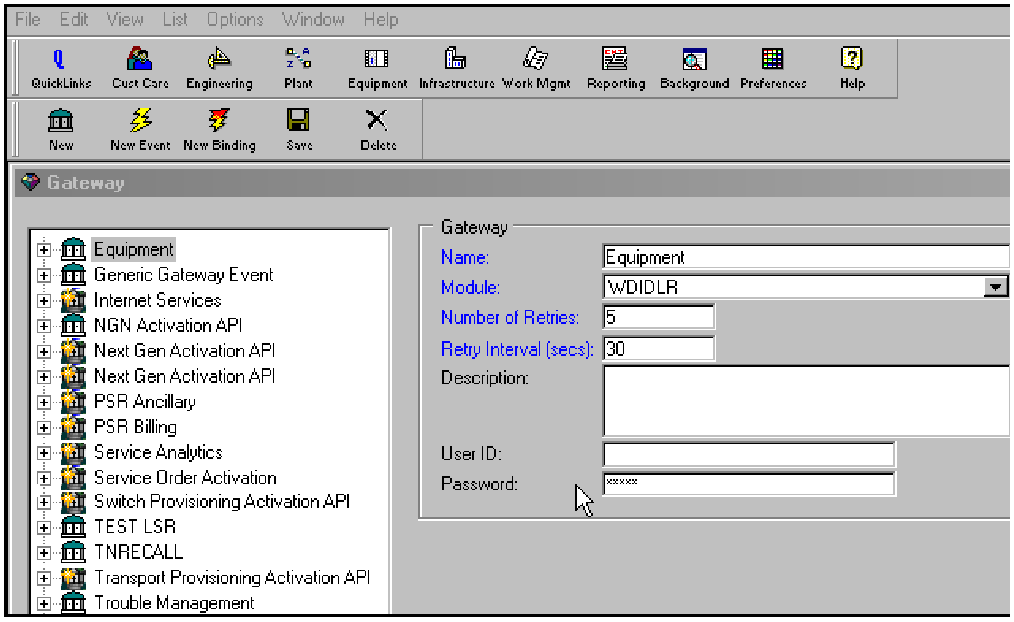
   The **Service Name** should be the same as the name of the BEA server that will receive events tied to the gateway.

3. Associate the gateway event with a task on the desired request.

For complete Help information on gateway events

1. Open the Gateway window in MetaSolv Solution.

To do this, click **Work Management** on the main toolbar, then click **Gateways** on the Work Management toolbar that appears.



Equipment
Generic Gateway Event
Internet Services
NGN Activation API
Next Gen Activation API
Next Gen Activation API
PSR Ancillary
PSR Billing
Service Analytics
Service Order Activation
Switch Provisioning Activation API
TEST LSR
TNRECALL
Transport Provisioning Activation API
Trouble Management

Gateway

Name:	Equipment
Module:	WDIDLR
Number of Retries:	5
Retry Interval (secs):	30
Description:	
User ID:	
Password:	xxxxx

2. Press **F1** for Help.

   The Help window that appears is for the Gateways window. You will find a number of links on this window that explain gateways and how to create them.

Configuring the gateway.ini file

You must make sure the Integration server is configured in the gateway.ini file. This is a configuration file for MetaSolv Solution and it is located on the machine running the MetaSolv Solution application server. The file can be found in the \appserver\gateway directory.

Use an ASCII editor to open the gateway.ini file. Make sure the INTEGRATIONSERVER line located in the ThreadProcs section is uncommented. If INTEGRATIONSERVER is commented, uncomment it and save the changes.

The following sample shows an uncommented INTEGRATIONSERVER line (in bold typeface).

```
[ThreadProcs]
INTEGRATIONSERVER=com.mslv.integration.integrationServer.S3Startup
EVENTPROC=MetaSolv.eventServer.S3Startup
EVENT2PROC=MetaSolv.event2Server.Event2ServerStartup
SYSTEMTASKSERVERPROC=com.mslv.core.api.internal.WM.systemTaskServer.
```

```
      SystemTaskServer
SIGNALSERVERPROC=com.metasolv.system.StartServer INTERNET_SIGNAL_SER
      VER=MetaSolv.CORBA.WDIINTERNETSERVICES.WDIRoot,MetaSolv.Sig
      nalServer.WDIInternetSignalServerRootImpl
```

Troubleshooting

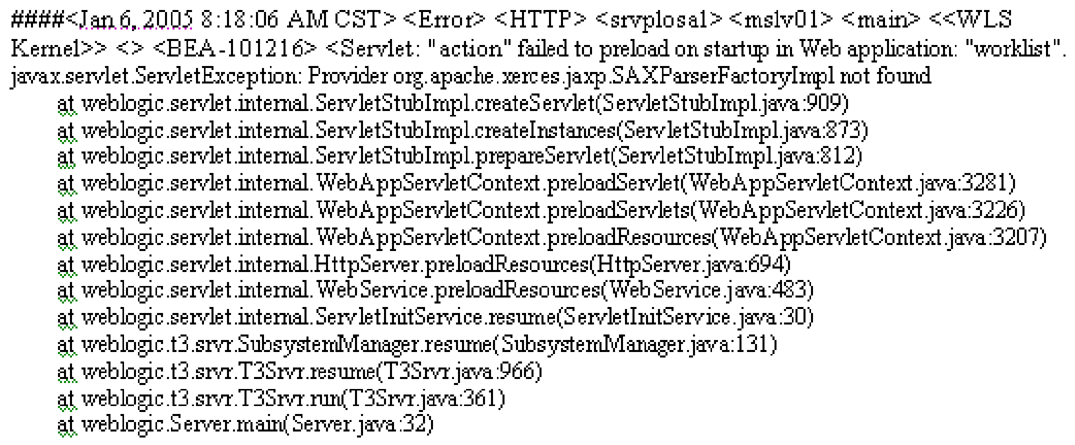
This chapter provides the information on troubleshooting servers, JDBC connections and error messages.

Server startup error

Problem: A server startup error is logged in the domain/server.log file when the mss_samples.ear application is running.

Solution: None. The problem is benign

The error is shown in the following code sample:



```
####<Jan 6, 2005 8:18:06 AM CST> <Error> <HTTP> <srvplosal> <mslv01> <main> <<WLS
Kernel>> <> <BEA-101216> <Servlet: "action" failed to preload on startup in Web application: "worklist".
javax.servlet.ServletException: Provider org.apache.xerces.jaxp.SAXParserFactoryImpl not found
      at weblogic.servlet.internal.ServletStubImpl.createServlet(ServletStubImpl.java:909)
      at weblogic.servlet.internal.ServletStubImpl.createInstances(ServletStubImpl.java:873)
      at weblogic.servlet.internal.ServletStubImpl.prepareServlet(ServletStubImpl.java:812)
      at weblogic.servlet.internal.WebAppServletContext.preloadServlet(WebAppServletContext.java:3281)
      at weblogic.servlet.internal.WebAppServletContext.preloadServlets(WebAppServletContext.java:3226)
      at weblogic.servlet.internal.WebAppServletContext.preloadResources(WebAppServletContext.java:3207)
      at weblogic.servlet.internal.HttpServer.preloadResources(HttpServer.java:694)
      at weblogic.servlet.internal.WebService.preloadResources(WebService.java:483)
      at weblogic.servlet.internal.ServletInitService.resume(ServletInitService.java:30)
      at weblogic.t3.srvr.SubsystemManager.resume(SubsystemManager.java:131)
      at weblogic.t3.srvr.T3Srvr.resume(T3Srvr.java:966)
      at weblogic.t3.srvr.T3Srvr.run(T3Srvr.java:361)
      at weblogic.Server.main(Server.java:32)
```

JRE update failure

Problem: In a Windows environment, a JRE update failure occurs during installation when multiple servers share the same BEA_HOME directory.

Cause: Another server(s) is running and using the files needed for installation.

Solution: Shut down the servers and jorbd processes (java) and rerun the installation program on the server where the failure occurred, then restart all servers.

DEBUG_PORT

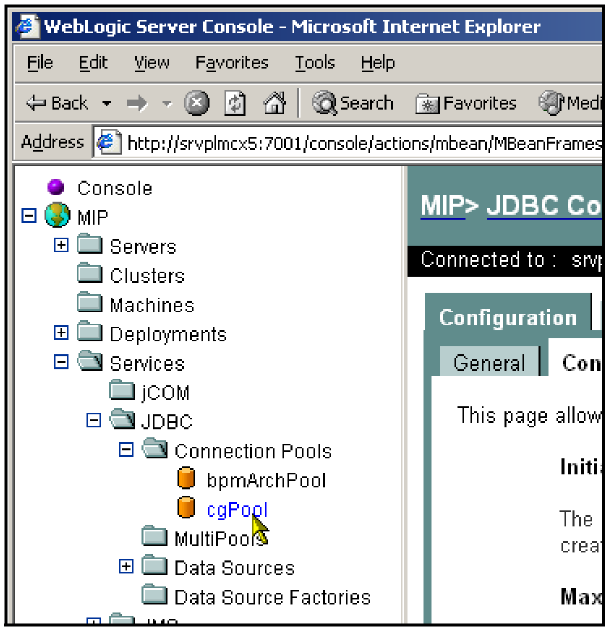
Problem: The default for this port is the value 8453 and currently it cannot be changed. This occurs only in development mode.

Solution: Currently none. The issue is being worked with BEA.

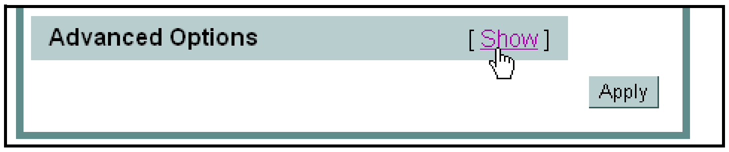
Testing JDBC connections

Generally, you should leave the JDBC connections settings on their default values. However, if you experience connection issues, there are some advanced options you can set to allow you to test your JDBC connections. To access these advanced options, do the following:

1. Log in to the WebLogic Server Console as described in "" on page 5.
2. In the treeview on the left, navigate to **MIP>Services>JDBC>Connection Pools>poolname** where poolname is the name of the connection pool.



3. Select the Configuration tab.
4. Select the Connections tab.
5. At the bottom of the Connections tab, click the **Show** link next to **Advanced Options**.



Use these options to test connections, change timeouts, and so on.

These options can have an impact on performance. After you have finished testing your connections, set these options back to their default values to maintain a higher performance level.

For more information about these advanced options, refer to the BEA documentation at http://e-docs.bea.com.

Firewall closes idle connections

If you have configured a firewall between the database and WebLogic Server, and this firewall closes idle connections after a certain amount of time, the JDBC pool refresh functionality can be used to ensure that connections from the pool are not closed by the firewall. A common error message thrown after such a closed connection is used follows:

```
java.sql.SQLException: ORA-03113: end-of-file on communication channel
at weblogic.db.oci.OciCursor.getCDAException(OciCursor.java:240)
at weblogic.jdbc.oci.Statement.executeQuery(Statement.java:916)
at ...
```

This error occurs because the socket connection is considered okay from both the WebLogic Server and the database side. So both may try to write into this socket connection and fail, because it has been closed by the firewall without notification or error message to the participating parties. Please use the refresh functionality to ensure that the connections are not idle long enough for the firewall to close them.

Configuring refresh functionality can be done by setting the `RefreshMinutes` property so that connections are tested at least one time during the idle period. To enable the refresh functionality, `TestTableName` property also has to be set. For more information, see: http://e-docs.bea.com/wls/docs81/config_xml/JDBCConnectionPool.html#RefreshMinutes

However, every JMS server takes one connection from the JDBC pool if a JDBC store is defined. This connection is considered as reserved by the pool, so that the refresh functionality will not test and refresh those connections. A typical error message would be:

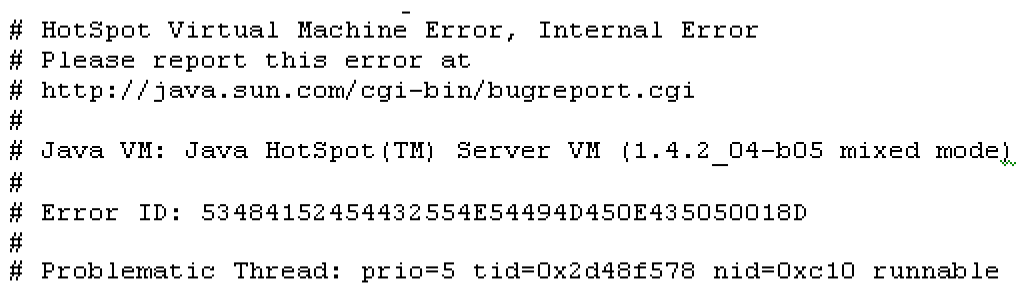
JMSServer "myJMSServer", store failure while writing message for queue myQueue, java.io.IOException

This kind of situation can be solved by either one of the following options:

- Send at least one dummy JMS message during the idle period, so that the firewall will not close the connection
- Disable the connection closure by the firewall
- Define a separate JDBC pool that will be used as JDBC store for JMS servers and use weblogic.Admin RESET_POOL to reopen the connections at least one time during the idle period

This problem has been addressed in later WebLogic Server versions (WLS 6.1 SP7, WLS 7.0 SP3, and WLS 8.1 SP1), so that if JMS is idle, the database is pinged every 5 minutes to keep the connection fresh and prevent the firewall from closing these connections

Java Hot Spot Error



```
# HotSpot Virtual Machine Error, Internal Error
# Please report this error at
# http://java.sun.com/cgi-bin/bugreport.cgi
#
# Java VM: Java HotSpot(TM) Server VM (1.4.2_04-b05 mixed mode)
#
# Error ID: 53484152454432554E54494D450E435050018D
#
# Problematic Thread: prio=5 tid=0x2d48f578 nid=0xc10 runnable
```

This error is documented in BEA 8.1 SP3 Release Notes. See the release notes at the following Web address:

http://e-docs.bea.com/wls.docss81/notes/issues.html

Cause: The following combination of Java options seem to cause the problem:

- Domain configured with Sun JDK 1.4.2_04 (not JRockit)
- Server running in production mode (**-server** option)
- Debugging enabled
- JSP Precompilation enabled (The weblogic.xml file for a Web application specifies precompile=true)

Workaround:

- Edit the server start script to start with -client (not -server) when debugging with the Sun JDK.
- Disable debugging (edit setDomainEnv or manually remove debug JVM arguments)
- Do not precompile when debugging.

Table errors

If you are getting table or view errors after deploying to production, read the following sections to make sure you have properly deployed your application.

- **Development mode and production mode**

  When you are developing, deploying and testing an application with WebLogic Workshop, the instance of WebLogic Server you are deploying to runs by default in *development mode.* In development mode, WebLogic Server behaves in ways that make it easier to iteratively develop and test an application: it automatically deploys the current application in an exploded format, server resources such as database tables and JMS queues necessary for the application to run are automatically created, and so on.

When the development cycle is complete, and the application is ready for use, you deploy it to an instance (or instances) of WebLogic Server running in *production* mode. In production mode applications are not automatically deployed and the server resources necessary for running an application are not automatically generated.

- **Manual Creation of Server Resources**

  When deploying EAR files to a production server, a certain amount of manual resource creation is necessary. When an application is built in an EAR file, a wlw-manifest.xml file is produced and placed in the application's META-INF directory. This file lists the JMS queues and database tables that need to be manually created on the target WebLogic Server for the application to run properly.

  When you are developing and testing an application with WebLogic Workshop, the creation of the necessary JMS queues and datatables on WebLogic Server takes place automatically on demand.

  Required database tables are indicated by a **<con:conversation-state-table />** tag. These tables are used by web services to store conversational state. For each occurrence of the <con:conversation-state-table /> tag in the wlw-manifest.xml file, you must create a corresponding data table on WebLogic Server.

  Required JMS queues are indicated by pairs of **<con:async-request-queue>** and **<con:async-request-error-queue>** tags. For each occurrence of these tags in the wlw-manifest.xml file, you must create a corresponding JMS queue on WebLogic Server *and* you must associate the members of the pair by referencing the <con:async-request-error-queue> in the ErrorDestination attribute of the <con:async-request-queue>.

  Optionally, you may want to enforce role restrictions on any controls that receive external callbacks. Controls that can receive external callbacks are indicated within a **<con:external-callbacks/>** tag in the wlw-manifest.xml file. Since the compilation process turns control files into individual methods on an EJB, you enforce the role restrictions on these post-compilation EJB methods.

Error message

Problem: The MetaSolv adapter returns the following message to the client during a XML API invocation when the Metasolv Solution application server is down:

java.lang.Exception: Error connecting to Metasolv Solution Server

Below is a sample response.

```
<ns:clientResponse xmlns:ns="http://www.openuri.org/">
<inv:createEntityByValueException xmlns:inv="http://www.metasolv.com/MIP
   InventoryManagementAPI">
<inv:createException>
```

```
<com:message xmlns:com="http://java.sun.com/products/oss/xml/Common">
   java.lang.Exception: Error connecting to Metasolv Solution Server
      </com:message>
</inv:createException>
</inv:createEntityByValueException>
</ns:clientResponse>
```

Solution: If you receive this error, restart the MetaSolv Solution application server, then the Integration server.

6

Performance

This chapter contains information on improving performance when XML APIs are used with MetaSolv Solution.

Setting up to precompile workflows

The XML API process controls can be precompiled during server start up. Precompiling improves performance by reducing the amount of time needed to load a control into memory the first time it is run. Note that after controls are loaded, performance is the same for precompiled and non-precompiled systems. While using precompilation saves time loading controls for the first time, it does lengthen server startup time.

Prerequisites

Before you start the setup, verify the following prerequisites:

- initDisp.jar is in the server *library* directory and has an entry in the server classpath
- The PRE_CLASSPATH statement in the startMSLVsingle (.cmd or .sh) script has an entry pointing to the initDisp.jar file, as shown in the following example:

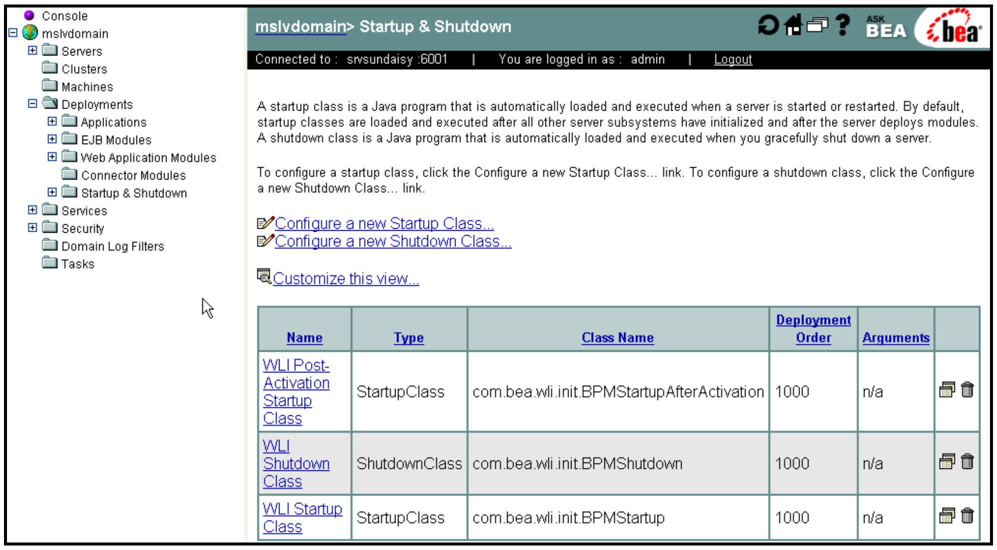
  PRE_CLASSPATH=%MSLV_LIBS_DIR%\initDisp.jar

To setup the pre-compilation

1. Log on to the WebLogic Server Administration console by typing the following URL address in the **Address** field of the Microsoft Internet Explorer:

   http://<*admin_host*:*port*>/console.
2. On the logon page, type your user ID and password, and press ENTER.
3. In the left pane, select **Deployments>Startup & Shutdown**.

The following pane appears on the right.



Console
mslvdomain
Servers
Clusters
Machines
Deployments
Applications
EJB Modules
Web Application Modules
Connector Modules
Startup & Shutdown
Services
Security
Domain Log Filters
Tasks

mslvdomain> Startup & Shutdown

Connected to : srvsundaisy :6001 | You are logged in as : admin | Logout

A startup class is a Java program that is automatically loaded and executed when a server is started or restarted. By default, startup classes are loaded and executed after all other server subsystems have initialized and after the server deploys modules. A shutdown class is a Java program that is automatically loaded and executed when you gracefully shut down a server.

To configure a startup class, click the Configure a new Startup Class... link. To configure a shutdown class, click the Configure a new Shutdown Class... link.

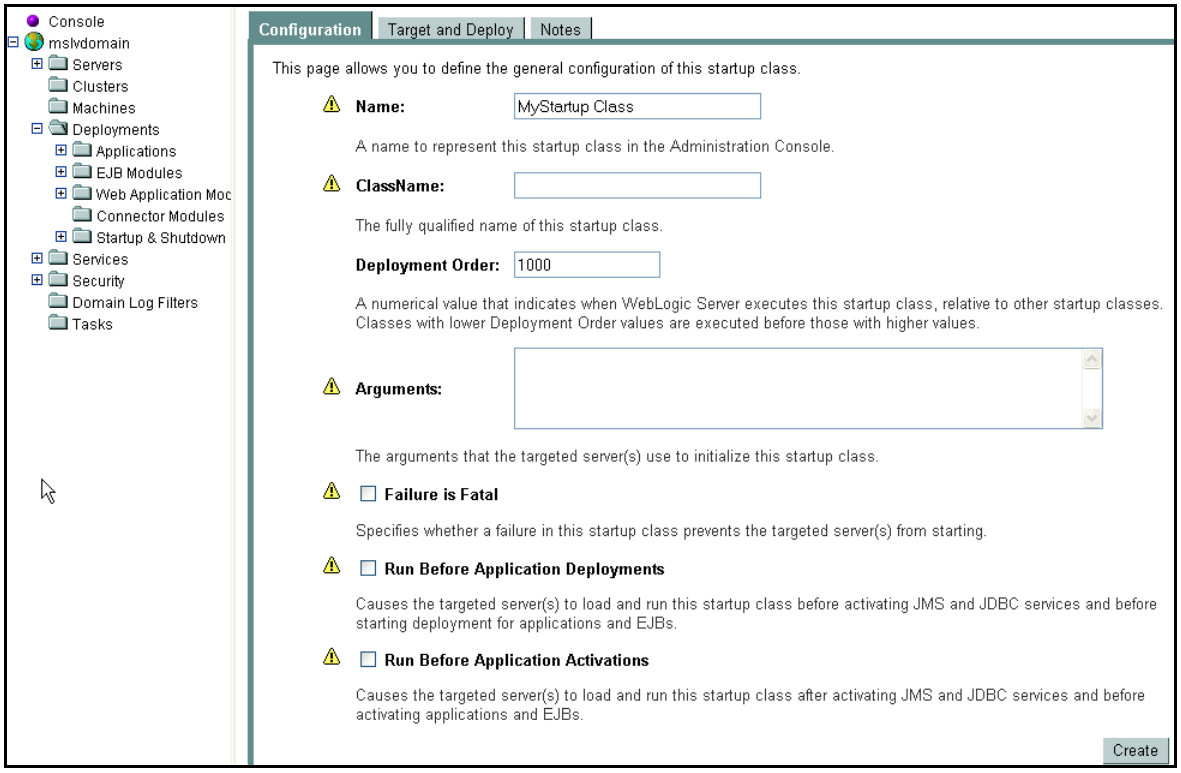
Configure a new Startup Class...
Configure a new Shutdown Class...

Customize this view...

Name	Type	Class Name	Deployment Order	Arguments	
WLI Post-Activation Startup Class	StartupClass	com.bea.wli.init.BPMStartupAfterActivation	1000	n/a	
WLI Shutdown Class	ShutdownClass	com.bea.wli.init.BPMShutdown	1000	n/a	
WLI Startup Class	StartupClass	com.bea.wli.init.BPMStartup	1000	n/a	

4. In the right pane, click **Configure a new Startup Class**.

   The following right pane appears.



Configuration | Target and Deploy | Notes

This page allows you to define the general configuration of this startup class.

Name: MyStartup Class

A name to represent this startup class in the Administration Console.

ClassName:

The fully qualified name of this startup class.

Deployment Order: 1000

A numerical value that indicates when WebLogic Server executes this startup class, relative to other startup classes. Classes with lower Deployment Order values are executed before those with higher values.

Arguments:

The arguments that the targeted server(s) use to initialize this startup class.

☐ **Failure is Fatal**

Specifies whether a failure in this startup class prevents the targeted server(s) from starting.

☐ **Run Before Application Deployments**

Causes the targeted server(s) to load and run this startup class before activating JMS and JDBC services and before starting deployment for applications and EJBs.

☐ **Run Before Application Activations**

Causes the targeted server(s) to load and run this startup class after activating JMS and JDBC services and before activating applications and EJBs.

Create

5. Enter the following information in the right pane and click **Create**:

   Name: DispCacheStartupClass

   ClassName: initializeDispCache

   Failure is Fatal: Select the check box.

6. On the Target and Deploy tab, select the server name to which the new class is to be deployed and click **Apply**.

   The new changes take effect the next time the server is started. To verify that the changes have occurred, check the log <*server_name*>.mss.log. The log will contain messages similar to the one shown here:

   /API/com/metasolv/api/workflow/customer/GetCustomerAccount/Sync.jpd

Appendix A: XML API sample code

MetaSolv provides a sample application, *mss_samples*, as a reference and guideline for developing Workshop applications to interface with Metasolv Solution XML APIs. If you are a developer setting up a workstation to do integration development, this application is designed to help you understand and work with the XML APIs.

Each sample is a test of a method included in the XML APIs. The sample code is placed in the following directory names: *customer, events, order,* and *inventory* so that you can quickly find the method you want.

The samples demonstrate the following information:

- Initialization
- Error handling
- Asynchronous interaction
- Http transmission
- JMS transmission

Where to find the sample files

The sample application is provided in the jar file *mss_xml_apiR#_b#.jar*

where:

R# is the release version

b# is the build version

For example: mss_xml_api_R603_b185.jar

The file contains the following entries:

- **mss_samples.jar**—This file contains the code, libraries and other files required for BEA 8.1 SP5 Workshop-based development for the mss_samples application.
- **mss_samples.ear**—This is the representation of an ear file that results from a successful build of the application.
- **ReleaseNotes.doc**—This contains any release specific notes, including enhancements and fixes.

During installation, the jar file containing the sample code is stored on the application server in the following location:

*METASOLV_HOME/SERVER_NAME/*appserver/samples

You can extract the contents of the jar file and place it in any location. The following procedure shows how to set up a sample application from the *samples* directory in a Windows environment.

Setting up the sample code

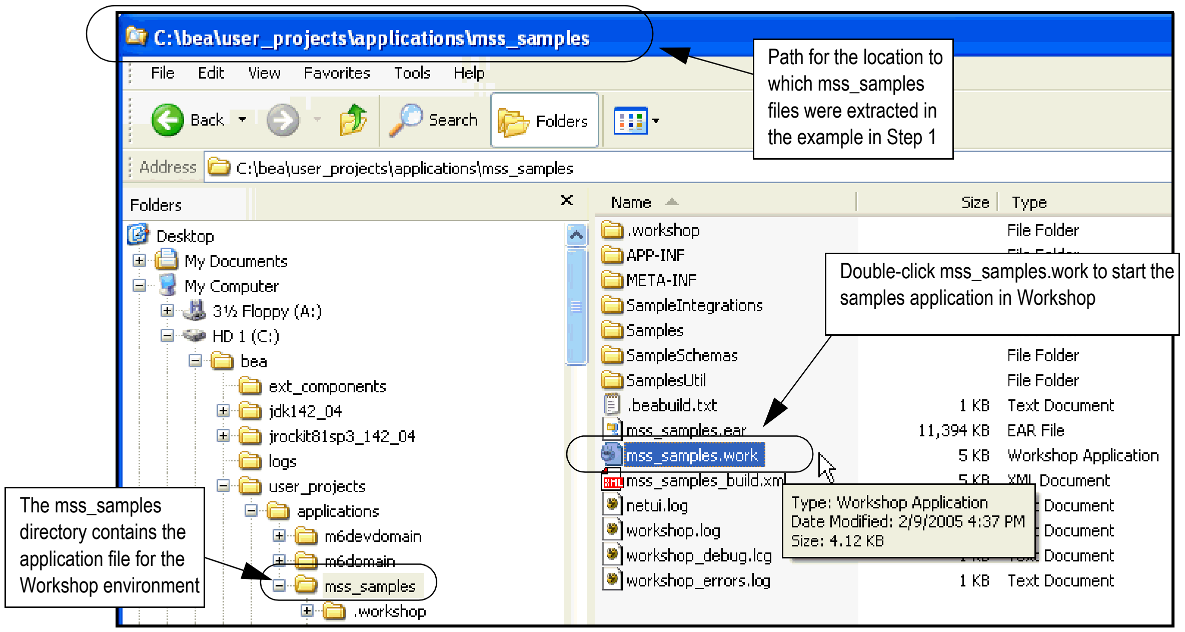
To set up the XML API sample code on a Windows workstation

1. Extract the mss_samples.jar into a directory.

   The directory can be an existing one or you can create a new directory. For example: BEA_HOME\user_projects\applications\mss_samples.

2. To open the application in your Workshop IDE, locate *mss_samples.work* in the mss_samples directory and double-click the file name.

   The following figure shows how the file structure appears in Microsoft Internet Explorer.



Path for the location to which mss_samples files were extracted in the example in Step 1

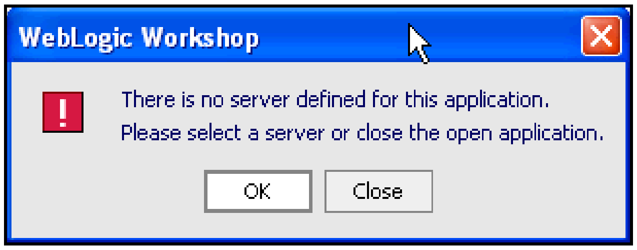
Double-click mss_samples.work to start the samples application in Workshop

The mss_samples directory contains the application file for the Workshop environment

Name	Size	Type
.workshop		File Folder
APP-INF		
META-INF		
SampleIntegrations		
Samples		
SampleSchemas		File Folder
SamplesUtil		File Folder
.beabuild.txt	1 KB	Text Document
mss_samples.ear	11,394 KB	EAR File
mss_samples.work	5 KB	Workshop Application
mss_samples_build.xml	5 KB	XML Document
netui.log		Document
workshop.log		Document
workshop_debug.log		Document
workshop_errors.log	1 KB	Text Document

Type: Workshop Application
Date Modified: 2/9/2005 4:37 PM
Size: 4.12 KB

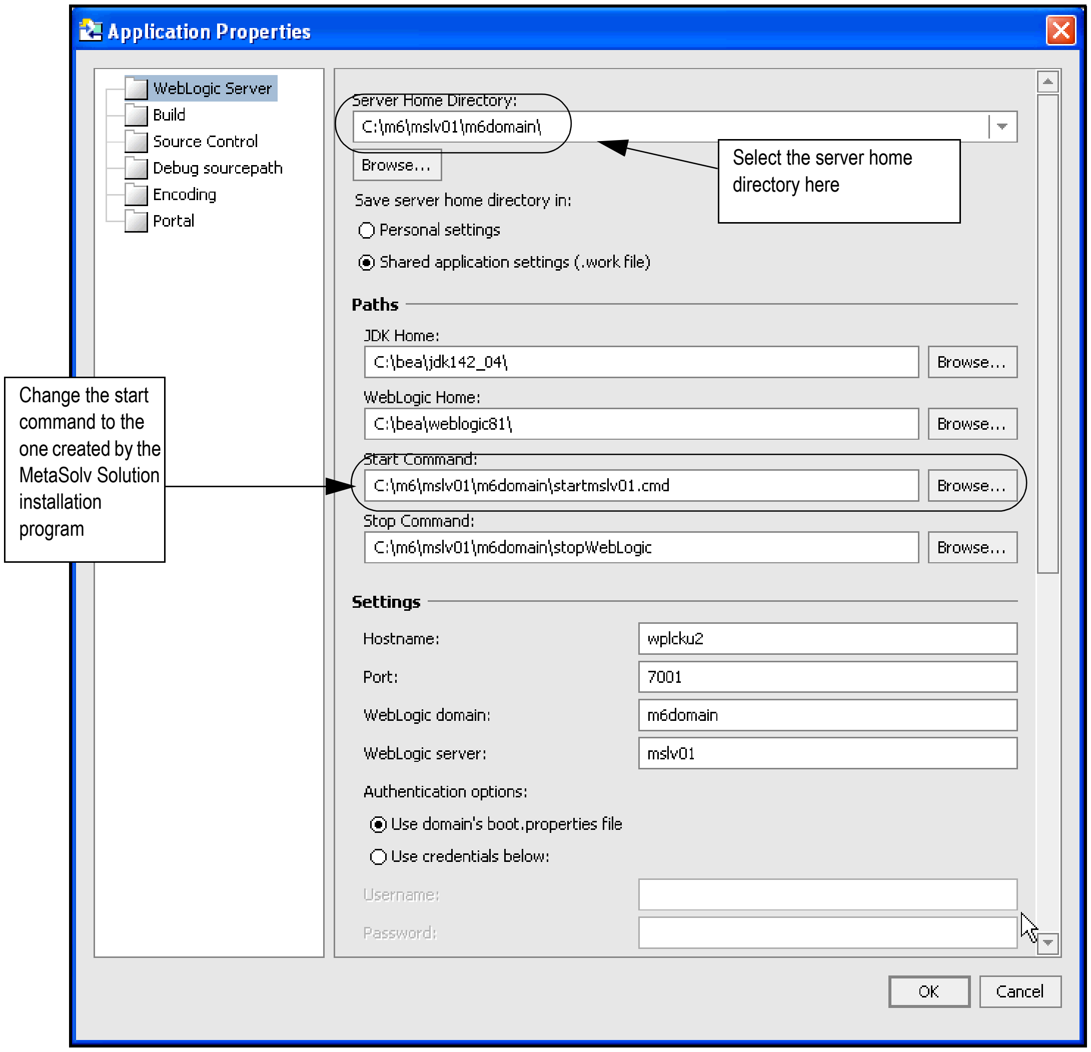
When you double-click mss_samples.work, the WebLogic Workshop application opens and the WebLogic Workshop window appears.


WebLogic Workshop

There is no server defined for this application.
Please select a server or close the open application.

3. Click **OK** to select a server and domain for the sample application.

   The Application Properties window appears.


Application Properties

Server Home Directory: C:\m6\mslv01\m6domain\

Browse...

Select the server home directory here

Save server home directory in:
( ) Personal settings
(●) Shared application settings (.work file)

Paths

JDK Home: C:\bea\jdk142_04\

WebLogic Home: C:\bea\weblogic81\

Start Command: C:\m6\mslv01\m6domain\startmslv01.cmd

Change the start command to the one created by the MetaSolv Solution installation program

Stop Command: C:\m6\mslv01\m6domain\stopWebLogic

Settings

Hostname: wplcku2
Port: 7001
WebLogic domain: m6domain
WebLogic server: mslv01

Authentication options:
(●) Use domain's boot.properties file
( ) Use credentials below:

Username:
Password:

4. Complete the following tasks on the window.
   a. Select the server home directory.

      If the server and the domain are on the same machine, remaining fields are populated automatically. MetaSolv recommends keeping the server and the domain on the same machine in a development environment.
   b. Change the server startup file to indicated the custom startup file created by MetaSolv Solution during the installation process.
   c. Accept the default values for the remaining fields on the window and click **OK**.

   When the server is running and available, a green icon like the one shown in the following figure appears on the Workshop status bar.

5. In Workshop's right pane, select the highest level available in the directory tree and select **Build Application**.

   This process can take from 15 to 45 minutes, depending on the size of the machine you are building the application on. When the build process is finished, a message appears in the Build tab indicating the build was successful.
6. When the build is complete, restart the application server.

Execute the workflows or browse through the workflows to see how they are constructed.

Upgrading sample files

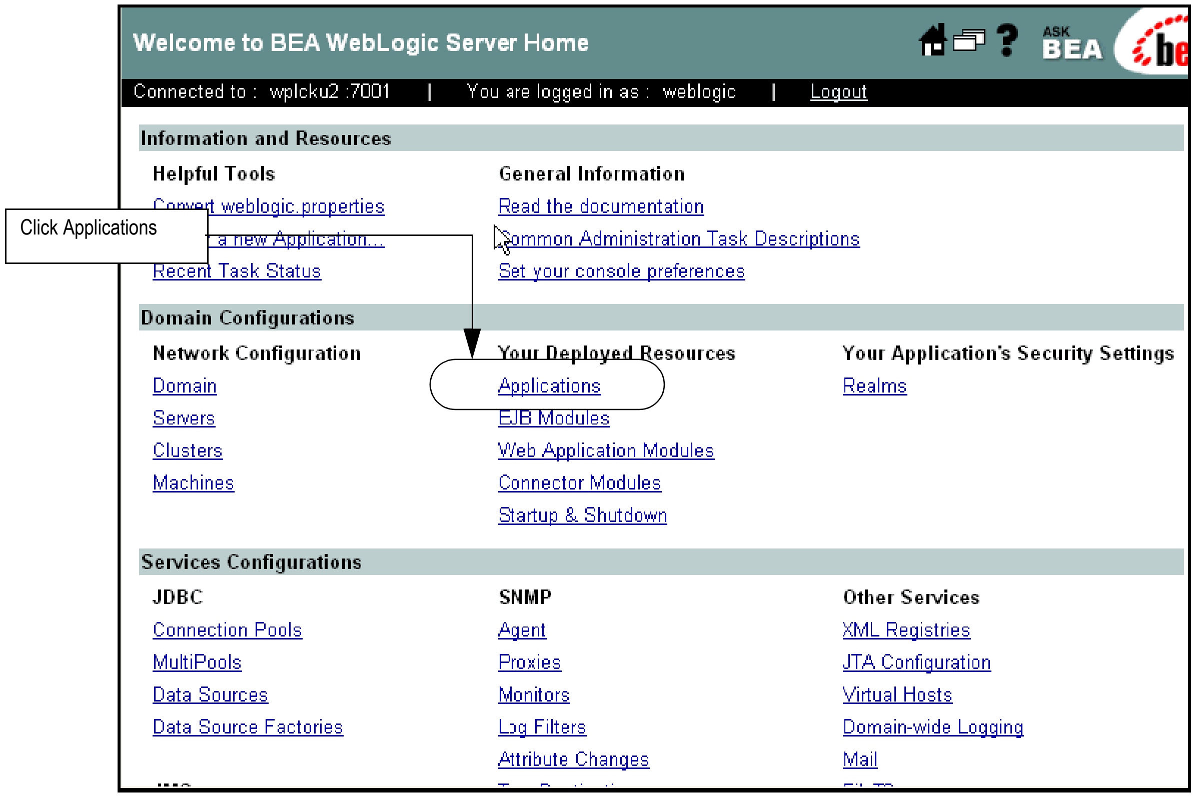
When you move from one release to another, you must also upgrade the sample file.

To upgrade a sample file

1. Upgrade the MetaSolv Solution core application.

   See the *MetaSolv Solution 6.0.x Setup Guide XML API Option* for instructions.
2. Log on to the WebLogic Server Administration Console.

3. In the right pane, click **Applications**.



Welcome to BEA WebLogic Server Home

Connected to : wplcku2 :7001 | You are logged in as : weblogic | Logout

Information and Resources

Helpful Tools	General Information
Convert weblogic.properties	Read the documentation
a new Application...	Common Administration Task Descriptions
Recent Task Status	Set your console preferences

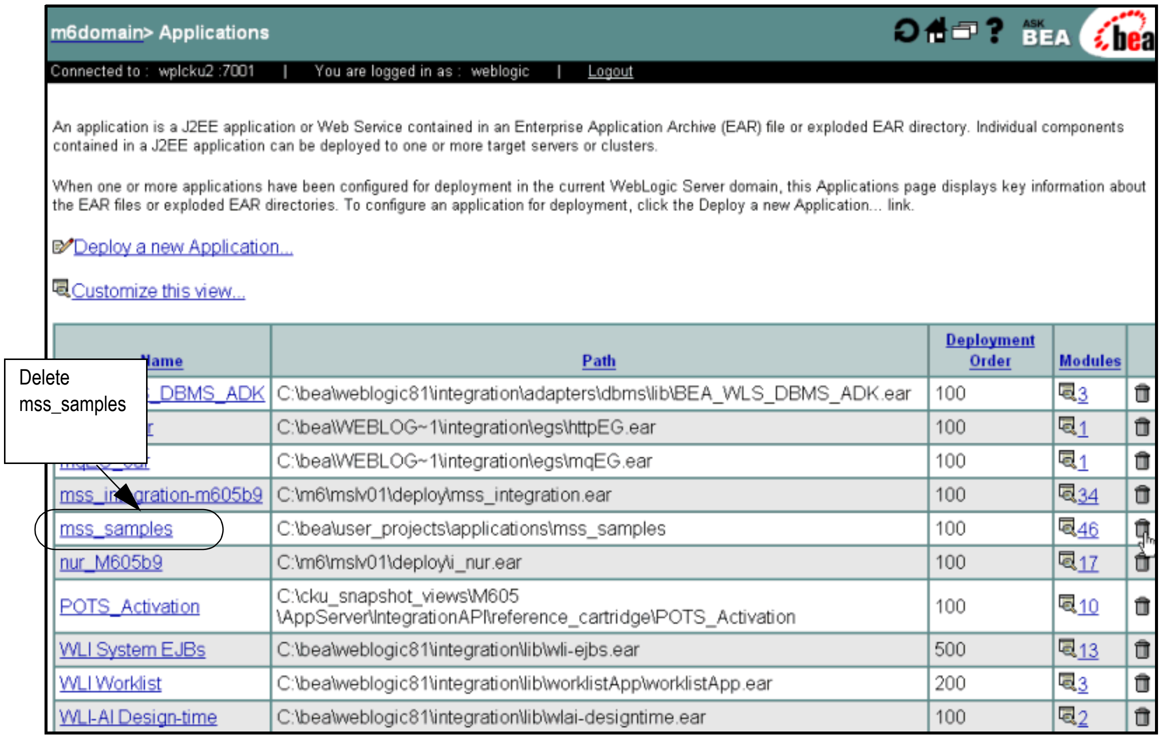
Domain Configurations

Network Configuration	Your Deployed Resources	Your Application's Security Settings
Domain	Applications	Realms
Servers	EJB Modules	
Clusters	Web Application Modules	
Machines	Connector Modules	
	Startup & Shutdown	

Services Configurations

JDBC	SNMP	Other Services
Connection Pools	Agent	XML Registries
MultiPools	Proxies	JTA Configuration
Data Sources	Monitors	Virtual Hosts
Data Source Factories	Log Filters	Domain-wide Logging
	Attribute Changes	Mail

When you click Applications, the following information appears in the right pane.



Connected to : wplcku2 :7001 | You are logged in as : weblogic | Logout

An application is a J2EE application or Web Service contained in an Enterprise Application Archive (EAR) file or exploded EAR directory. Individual components contained in a J2EE application can be deployed to one or more target servers or clusters.

When one or more applications have been configured for deployment in the current WebLogic Server domain, this Applications page displays key information about the EAR files or exploded EAR directories. To configure an application for deployment, click the Deploy a new Application... link.

Deploy a new Application...

Customize this view...

Name	Path	Deployment Order	Modules	
s_DBMS_ADK	C:\bea\weblogic81\integration\adapters\dbms\lib\BEA_WLS_DBMS_ADK.ear	100	3	🗑
r	C:\bea\WEBLOG~1\integration\egs\httpEG.ear	100	1	🗑
mqEG_ear	C:\bea\WEBLOG~1\integration\egs\mqEG.ear	100	1	🗑
mss_integration-m605b9	C:\m6\mslv01\deploy\mss_integration.ear	100	34	🗑
mss_samples	C:\bea\user_projects\applications\mss_samples	100	46	🗑
nur_M605b9	C:\m6\mslv01\deploy\i_nur.ear	100	17	🗑
POTS_Activation	C:\cku_snapshot_views\M605 \AppServer\IntegrationAPI\reference_cartridge\POTS_Activation	100	10	🗑
WLI System EJBs	C:\bea\weblogic81\integration\lib\wli-ejbs.ear	500	13	🗑
WLI Worklist	C:\bea\weblogic81\integration\lib\worklistApp\worklistApp.ear	200	3	🗑
WLI-AI Design-time	C:\bea\weblogic81\integration\lib\wlai-designtime.ear	100	2	🗑

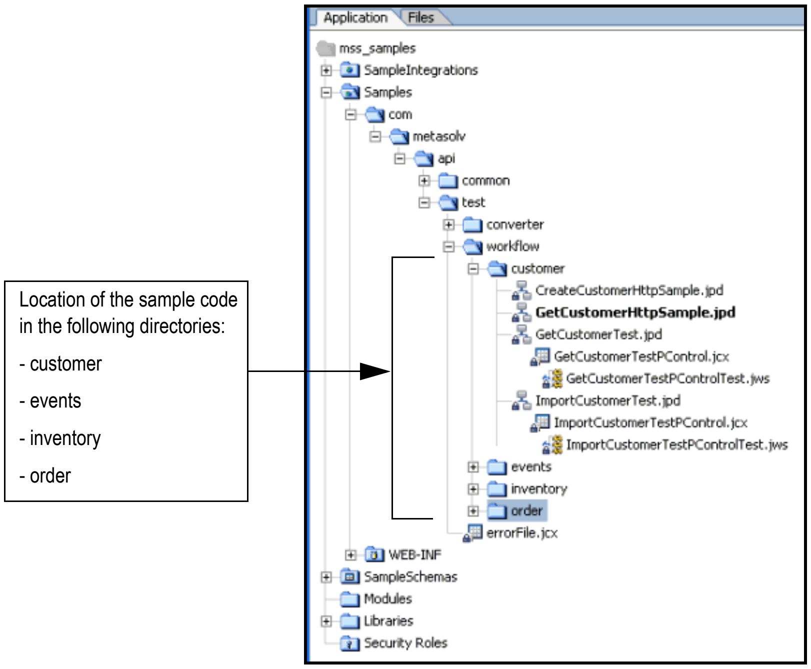
4. Delete mss_samples by dragging it to the trash can on the right.
5. Refresh the screen to ensure that the deletion occurred.
6. Repeat the steps in the previous procedure to add the new sample application.

Viewing the samples in Workshop

To view the samples in Workshop

1. Navigate to the directory where the you extracted the files from *mss_samples.jar*.

2. Follow the directory path shown in the following figure.

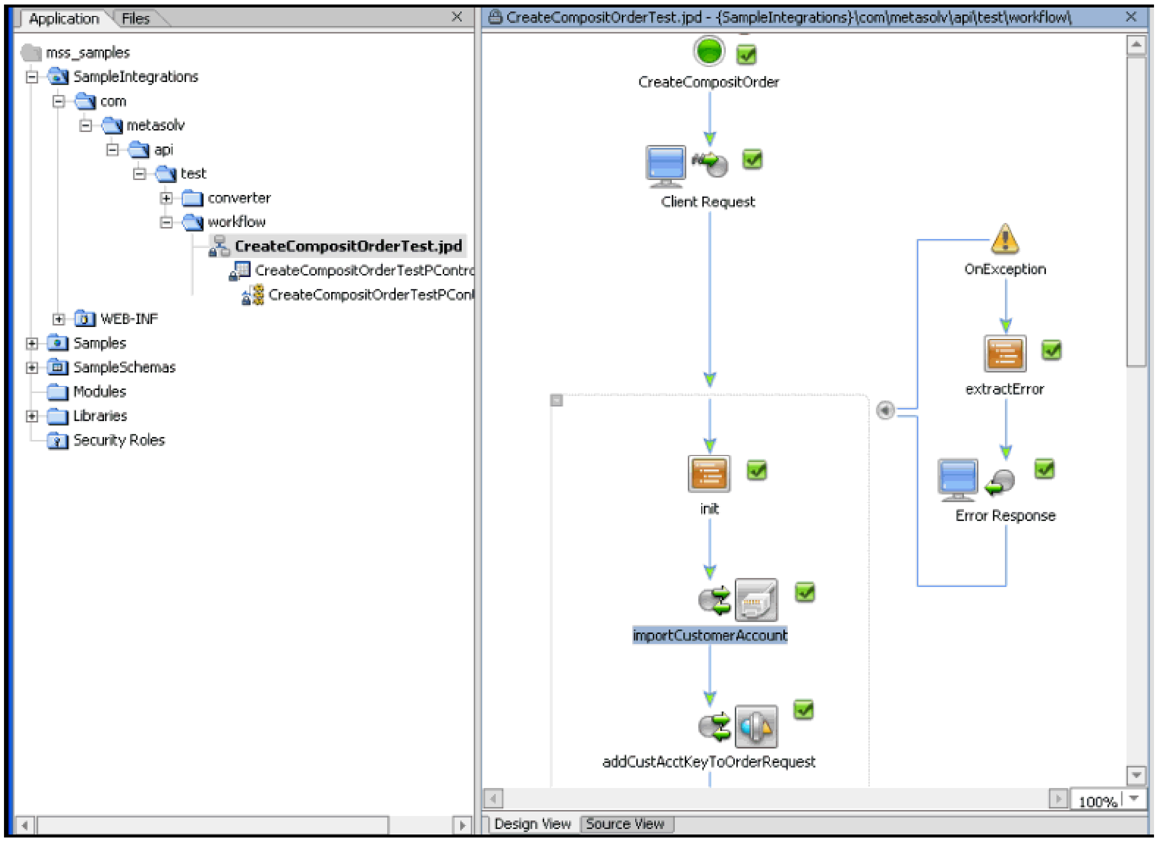


3. Double-click a .jpd file under one of the following directories: customer, events, inventory, order.

   The workflow appears on the Workshop canvas.

Composite sample

A composite order is also included in the samples. The composite order combines multiple methods to get a desired result. The following figure shows where in the directory structure the composite order sample is located and how the workflow looks displayed in Design View.



Appendix B: Control interfaces

Customer Management control interface

```
package com.metasolv.api.control;


import com.bea.control.Control;


public interface CustomerManagement extends Control
{
com.metasolv.mip.customerManagementAPI.UpdateCustomerAccountByValueResponseD
ocument
importCustomerAccount(com.metasolv.mip.customerManagementAPI.UpdateCustomerA
ccountByValueRequestDocument request);


com.metasolv.mip.customerManagementAPI.GetCustomerAccountByKeyResponseDocume
nt
getCustomerAccountByKey(com.metasolv.mip.customerManagementAPI.GetCustomerAc
countByKeyRequestDocument request);



com.metasolv.mip.customerManagementAPI.DeleteCustomerAccountByKeyResponseDoc
ument
deleteCustomerRequest(com.metasolv.mip.customerManagementAPI.DeleteCustomerA
ccountByKeyRequestDocument x0);
}
```

Order Management control interface

```
package com.metasolv.api.control;

import com.bea.control.Control;

public interface OrderManagement extends Control
{
com.metasolv.mip.orderManagementAPI.QueryOrderManagementResponseDocument
queryOrderManagementRequest(com.metasolv.mip.orderManagementAPI.QueryOrderMa
nagementRequestDocument queryOrderRequest);

com.metasolv.mip.orderManagementAPI.StartOrderByKeyResponseDocument
startOrderByKeyRequest(com.metasolv.mip.orderManagementAPI.StartOrderByKeyRe
questDocument request);

com.metasolv.mip.orderManagementAPI.UpdateOrderManagementResponseDocument
updateOrderManagementRequest(com.metasolv.mip.orderManagementAPI.UpdateOrder
ManagementRequestDocument orderRequest);

com.metasolv.mip.orderManagementAPI.GetOrderByKeyResponseDocument
getOrderByKeyRequest(com.metasolv.mip.orderManagementAPI.GetOrderByKeyReques
tDocument orderRequest);

com.metasolv.mip.orderManagementAPI.CreateOrderByValueResponseDocument
createOrderByValueRequest(com.metasolv.mip.orderManagementAPI.CreateOrderByV
alueRequestDocument orderRequest);

com.metasolv.mip.orderManagementAPI.AssignProvisionPlanProcedureResponseDocu
ment
assignProvisionPlanProcedureRequest(com.metasolv.mip.orderManagementAPI.Assi
gnProvisionPlanProcedureRequestDocument request);

com.metasolv.mip.orderManagementAPI.GetActivationDataByKeyResponseDocument
getActivationDataByKeyRequest(com.metasolv.mip.orderManagementAPI.GetActivat
ionDataByKeyRequestDocument x0);

com.metasolv.mip.orderManagementAPI.TransferTaskResponseDocument
transferTaskRequest(com.metasolv.mip.orderManagementAPI.TransferTaskRequestD
ocument x0);
```

```
com.metasolv.mip.orderManagementAPI.UpdateE911DataReponseDocument
updateE911DataRequest(com.metasolv.mip.orderManagementAPI.UpdateE911DataRequ
estDocument updateE911RequestDocument);


com.metasolv.mip.orderManagementAPI.GetE911DataResponseDocument
getE911DataRequest(com.metasolv.mip.orderManagementAPI.GetE911DataRequestDoc
ument e911DataRequest);


com.metasolv.mip.orderManagementAPI.UpdateEstimatedCompletionDateResponseDoc
ument
updateEstimationCompletedDateRequest(com.metasolv.mip.orderManagementAPI.Upd
ateEstimatedCompletionDateRequestDocument
updateEstimationCompletedDateRequestDocument);


com.metasolv.mip.orderManagementAPI.AddTaskJeopardyResponseDocument
addTaskJeopardyRequest(com.metasolv.mip.orderManagementAPI.AddTaskJeopardyRe
questDocument addTaskJeopardyRequestDocument);


com.metasolv.mip.orderManagementAPI.GetTaskDetailResponseDocument
getTaskDetailRequest(com.metasolv.mip.orderManagementAPI.GetTaskDetailReques
tDocument getTaskDetailRequestDocument);


com.metasolv.mip.orderManagementAPI.GetTaskJeopardyResponseDocument
TaskJeopardyRequest(com.metasolv.mip.orderManagementAPI.GetTaskJeopardyReque
stDocument getTaskJeopardyRequest);


com.metasolv.mip.orderManagementAPI.GetPSROrderByTNResponseDocument
getPSROrderByTN(com.metasolv.mip.orderManagementAPI.GetPSROrderByTNRequestDo
cument getPSROrderByTNRequestDoc);


com.metasolv.mip.orderManagementAPI.ProcessSuppOrderResponseDocument
processSuppOrder(com.metasolv.mip.orderManagementAPI.ProcessSuppOrderRequest
Document x0);


com.metasolv.mip.orderManagementAPI.GetCNAMDataResponseDocument
getCnamDataRequest();


com.metasolv.mip.orderManagementAPI.GetLIDBDataResponseDocument
getLidbDataRequest(com.metasolv.mip.orderManagementAPI.GetLIDBDataRequestDoc
ument getLIDBDataRequestDoc);
```

```
com.metasolv.mip.orderManagementAPI.UpdateCNAMDataReponseDocument
updateCnamDataRequest(com.metasolv.mip.orderManagementAPI.UpdateCNAMDataRequ
estDocument updateCNAMDataRequestDoc);


com.metasolv.mip.orderManagementAPI.UpdateLIDBDataReponseDocument
updateLidbDataRequest(com.metasolv.mip.orderManagementAPI.UpdateLIDBDataRequ
estDocument updateLIDBDataRequestDoc);


com.metasolv.mip.orderManagementAPI.ReopenTaskResponseDocument
reopenTaskRequest(com.metasolv.mip.orderManagementAPI.ReopenTaskRequestDocum
ent reopenTaskRequestDoc);


com.metasolv.mip.orderManagementAPI.CreateAttachmentResponseDocument
createAttachment(com.metasolv.mip.orderManagementAPI.CreateAttachmentRequest
Document x0, com.bea.xml.XmlObject x1);


com.metasolv.mip.orderManagementAPI.CreateOrderRelationshipResponseDocument
createOrderRelationshipRequest(com.metasolv.mip.orderManagementAPI.CreateOrd
erRelationshipRequestDocument x0);

}
```

Inventory Management interface control

```
package com.metasolv.api.control;


import com.bea.control.Control;


public interface InventoryManagement extends Control
{
com.metasolv.mip.inventoryManagementAPI.CreateEntityByValueResponseDocument
createEntityByValueRequest(com.metasolv.mip.inventoryManagementAPI.CreateEntityBy
ValueRequestDocument request);


com.metasolv.mip.orderManagementAPI.GetServiceRequestDLRsResponseDocument
getServiceRequestDLRsValue(com.metasolv.mip.orderManagementAPI.GetServiceRequestD
LRsValueDocument request);


com.metasolv.mip.inventoryManagementAPI.GetEntityByKeyResponseDocument
getEntityByKeyRequest(com.metasolv.mip.inventoryManagementAPI.GetEntityByKeyReque
stDocument request);


com.metasolv.mip.inventoryManagementAPI.UpdateEntityByValueResponseDocument
updateEntityByValueRequest(com.metasolv.mip.inventoryManagementAPI.UpdateEntityBy
ValueRequestDocument request);


com.metasolv.mip.inventoryManagementAPI.QueryInventoryManagementResponseDocument
queryInventoryManagementRequest(com.metasolv.mip.inventoryManagementAPI.QueryInve
ntoryManagementRequestDocument queryInventoryRequest);


com.metasolv.mip.inventoryManagementAPI.UpdateTNResponseDocument
updateTNRequest(com.metasolv.mip.inventoryManagementAPI.UpdateTNRequestDocument
x0);


com.metasolv.mip.inventoryManagementAPI.ProcessTNRecallResponseDocument
tnRecall(com.metasolv.mip.inventoryManagementAPI.ProcessTNRecallRequestDocument
x0);


com.metasolv.mip.inventoryManagementAPI.ProcessTNValidationResponseDocument
tnValidationRequest(com.metasolv.mip.inventoryManagementAPI.ProcessTNValidationRe
questDocument processTNValidationRequestDocument);


java.lang.String
auditTrailRecording(com.metasolv.mip.inventoryManagementAPI.QueryInventoryManagem
entResponseDocument queryInventoryManagementResponseDoc, java.lang.String
partnerId, java.lang.String successOrFailure);
com.metasolv.mip.inventoryManagementAPI.GetNetworkAreasByGeoAreaResponseDocument
getNetworkAreasByGeoAreaRequest(com.metasolv.mip.inventoryManagementAPI.GetNetwor
kAreasByGeoAreaRequestDocument x0);

com.metasolv.mip.inventoryManagementAPI.GetNetworkComponentsResponseDocument
getNetworkComponentsRequest(com.metasolv.mip.inventoryManagementAPI.GetNetworkCom
ponentsRequestDocument x0);

com.metasolv.mip.inventoryManagementAPI.GetIpAddressesResponseDocument
getIpAddressesRequest(com.metasolv.mip.inventoryManagementAPI.GetIpAddressesReque
stDocument getIpAddressesReQuestDoc);

com.metasolv.mip.inventoryManagementAPI.CreateInventoryAssociationResponseDocumen
t
inventoryAssociationRequest(com.metasolv.mip.inventoryManagementAPI.CreateInvento
ryAssociationRequestDocument request);

com.metasolv.mip.inventoryManagementAPI.CreateNewInventoryItemResponseDocument
createNewInventoryItemRequest(com.metasolv.mip.inventoryManagementAPI.CreateNewIn
ventoryItemRequestDocument createNewInventoryItemRequestDoc);

com.metasolv.mip.inventoryManagementAPI.QueryNetworkLocationResponseDocument
queryNetworkLocation(com.metasolv.mip.inventoryManagementAPI.QueryNetworkLocation
RequestDocument queryNetworkLocationRequestDoc);

com.metasolv.mip.inventoryManagementAPI.QueryEndUserLocationResponseDocument
queryEndUserLocation(com.metasolv.mip.inventoryManagementAPI.QueryEndUserLocation
RequestDocument queryEndUserLocationRequestDoc);

com.metasolv.mip.inventoryManagementAPI.GetLocationResponseDocument
getLocationRequest(com.metasolv.mip.inventoryManagementAPI.GetLocationRequestDocu
ment request);

com.metasolv.mip.inventoryManagementAPI.DeleteLocationResponseDocument
deleteLocationRequest(com.metasolv.mip.inventoryManagementAPI.DeleteLocationByKey
Document deleteLocation);

com.metasolv.mip.inventoryManagementAPI.UpdateLocationResponseDocument
updateLocationRequest(com.metasolv.mip.inventoryManagementAPI.UpdateLocationReque
stDocument locationData);

com.metasolv.mip.inventoryManagementAPI.CreateLocationResponseDocument
createLocationRequest(com.metasolv.mip.inventoryManagementAPI.CreateLocationReque
stDocument request);
}
```

Network Resource Management interface control

```
package com.metasolv.api.control;


import com.bea.control.Control;


public interface NetworkResourceManagement extends Control

{

com.metasolv.mip.inventoryManagementAPI.GetAvailablePhysicalPortsResponseDoc
ument
getAvailablePhysicalPortsRequest(com.metasolv.mip.inventoryManagementAPI.Get
AvailablePhysicalPortsRequestDocument x0);

}
```

Service Order Activation interface control

```
package com.metasolv.api.control;


import com.bea.control.Control;


public interface ServiceOrderActivation extends Control

{

com.metasolv.mip.serviceOrderActivationAPI.CreateSOAMessageResponseDocument
createSoaMessageRequest(com.metasolv.mip.serviceOrderActivationAPI.CreateSOA
MessageRequestDocument x0);


com.metasolv.mip.serviceOrderActivationAPI.GetSOATNsForOrderResponseDocument
getSoaTnsForOrderRequest(com.metasolv.mip.serviceOrderActivationAPI.GetSOATN
sForOrderRequestDocument x0);


com.metasolv.mip.serviceOrderActivationAPI.GetSOADefaultsResponseDocument
getSoaDefaultsRequest(com.metasolv.mip.serviceOrderActivationAPI.GetSOADefau
ltsRequestDocument x0);


com.metasolv.mip.serviceOrderActivationAPI.GetSOAInformationResponseDocument
getSoaInformationRequest(com.metasolv.mip.serviceOrderActivationAPI.GetSOAIn
formationRequestDocument x0);


com.metasolv.mip.serviceOrderActivationAPI.GetSOAMessagesToSendResponseDocum
ent
getSoaMessageToSendRequest(com.metasolv.mip.serviceOrderActivationAPI.GetSOA
MessagesToSendRequestDocument x0);


com.metasolv.mip.serviceOrderActivationAPI.SetTNSOACompleteResponseDocument
setTnSoaCompleteRequest(com.metasolv.mip.serviceOrderActivationAPI.SetTNSOAC
ompleteRequestDocument x0);

}
```

Event Management control interface

```
package com.metasolv.api.control;


import com.bea.control.Control;


public interface EventManagement extends Control
{
java.lang.String
updateInboundEventStatus(com.metasolv.mip.orderManagementAPI.UpdateOrderTask
EventProcedureValueDocument x0);


com.metasolv.mip.orderManagementEvents.MetasolvOrderTaskEventStatusChangeDoc
ument getEventStatus(java.lang.String pExternalSystemName, java.lang.String
pExternalSystemKey);


java.lang.String
updateOutboundEventStatus(com.metasolv.mip.orderManagementEvents.MetasolvOrd
erTaskEventStatusChangeDocument pRequest);
}
```

LSR6 Management control interface

```
package com.metasolv.api.control;


import com.bea.control.Control;


public interface LSRManagement extends Control
{
com.metasolv.mip.lsr6API.GetLRByKeyResponseDocument
getLRByKeyRequest(com.metasolv.mip.lsr6API.GetLRByKeyRequestDocument x0);


com.metasolv.mip.lsr6API.GetDLByKeyResponseDocument
getDLByKeyRequest(com.metasolv.mip.lsr6API.GetDLByKeyRequestDocument x0);


com.metasolv.mip.lsr6API.GetLSRByKeyResponseDocument
getLSRByKeyRequest(com.metasolv.mip.lsr6API.GetLSRByKeyRequestDocument x0);


com.metasolv.mip.lsr6API.GetLSRCMByKeyResponseDocument
getLSRCMByKeyRequest(com.metasolv.mip.lsr6API.GetLSRCMByKeyRequestDocument
x0);


com.metasolv.mip.lsr6API.CreateDSCNByValueResponseDocument
createDSCNByValueRequest(com.metasolv.mip.lsr6API.CreateDSCNByValueRequestDo
cument x0);


com.metasolv.mip.lsr6API.CreateDSREDByValueResponseDocument
createDSREDByValueRequest(com.metasolv.mip.lsr6API.CreateDSREDByValueRequest
Document x0);


com.metasolv.mip.lsr6API.CreateLRByValueResponseDocument
createLRByValueRequest(com.metasolv.mip.lsr6API.CreateLRByValueRequestDocume
nt x0);


com.metasolv.mip.lsr6API.CreateLSRCMByValueResponseDocument
createLSRCMByValueRequest(com.metasolv.mip.lsr6API.CreateLSRCMByValueRequest
Document x0);


com.metasolv.mip.lsr6API.CreateNPLSRByValueResponseDocument
createNPLSRByValueRequest(com.metasolv.mip.lsr6API.CreateNPLSRByValueRequest
Document x0);
com.metasolv.mip.lsr6API.QueryCCNAResponseDocument
queryCCNARequest(com.metasolv.mip.lsr6API.QueryCCNARequestDocument x0);


com.metasolv.mip.lsr6API.QueryLSRResponseDocument
queryLSRRequest(com.metasolv.mip.lsr6API.QueryLSRRequestDocument x0);


com.metasolv.mip.lsr6API.QueryLSRForPONCCNAVERResponseDocument
queryLSRForPONCCNAVERRequest(com.metasolv.mip.lsr6API.QueryLSRForPONCCNAVERR
equestDocument x0);


com.metasolv.mip.lsr6API.QueryPONSForCCNAResponseDocument
queryPONSForCCNARequest(com.metasolv.mip.lsr6API.QueryPONSForCCNARequestDocu
ment x0);


com.metasolv.mip.lsr6API.CreateLSROrderByValueResponseDocument
createLSROrderByValueRequest(com.metasolv.mip.lsr6API.CreateLSROrderByValueR
equestDocument x0);
}
```

LSR9 Management control interface

```
package com.metasolv.api.lsr9.control;


import com.bea.control.Control;


public interface LSRManagement extends Control
{


com.metasolv.mip.lsr9API.GetLRByKeyResponseDocument
getLRByKeyRequest(com.metasolv.mip.lsr9API.GetLRByKeyRequestDocument x0);


com.metasolv.mip.lsr9API.GetDLByKeyResponseDocument
getDLByKeyRequest(com.metasolv.mip.lsr9API.GetDLByKeyRequestDocument x0);


com.metasolv.mip.lsr9API.GetLSRByKeyResponseDocument
getLSRByKeyRequest(com.metasolv.mip.lsr9API.GetLSRByKeyRequestDocument x0);


com.metasolv.mip.lsr9API.GetLSRCMByKeyResponseDocument
getLSRCMByKeyRequest(com.metasolv.mip.lsr9API.GetLSRCMByKeyRequestDocument
x0);


com.metasolv.mip.lsr9API.CreateDSCNByValueResponseDocument
createDSCNByValueRequest(com.metasolv.mip.lsr9API.CreateDSCNByValueRequestDo
cument x0);


com.metasolv.mip.lsr9API.CreateDSREDByValueResponseDocument
createDSREDByValueRequest(com.metasolv.mip.lsr9API.CreateDSREDByValueRequest
Document x0);


com.metasolv.mip.lsr9API.CreateLRByValueResponseDocument
createLRByValueRequest(com.metasolv.mip.lsr9API.CreateLRByValueRequestDocume
nt x0);


com.metasolv.mip.lsr9API.CreateLSRCMByValueResponseDocument
createLSRCMByValueRequest(com.metasolv.mip.lsr9API.CreateLSRCMByValueRequest
Document x0);


com.metasolv.mip.lsr9API.CreateNPLSRByValueResponseDocument
createNPLSRByValueRequest(com.metasolv.mip.lsr9API.CreateNPLSRByValueRequest
Document x0);
com.metasolv.mip.lsr9API.QueryCCNAResponseDocument
queryCCNARequest(com.metasolv.mip.lsr9API.QueryCCNARequestDocument x0);


com.metasolv.mip.lsr9API.QueryLSRResponseDocument
queryLSRRequest(com.metasolv.mip.lsr9API.QueryLSRRequestDocument x0);


com.metasolv.mip.lsr9API.QueryLSRForPONCCNAVERResponseDocument
queryLSRForPONCCNAVERRequest(com.metasolv.mip.lsr9API.QueryLSRForPONCCNAVERR
equestDocument x0);


com.metasolv.mip.lsr9API.QueryPONSForCCNAResponseDocument
queryPONSForCCNARequest(com.metasolv.mip.lsr9API.QueryPONSForCCNARequestDocu
ment x0);


com.metasolv.mip.lsr9API.CreateLSROrderByValueResponseDocument
createLSROrderByValueRequest(com.metasolv.mip.lsr9API.CreateLSROrderByValueR
equestDocument x0);
}
```

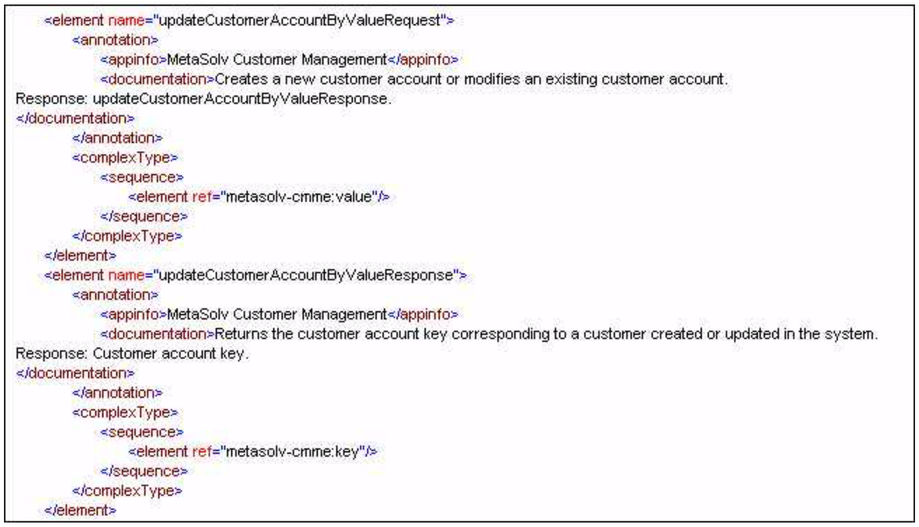
LSR10 Management control interface

```
package com.metasolv.api.lsr9.control;


import com.bea.control.Control;


public interface LSRManagement extends Control

{


com.metasolv.mip.lsr9API.GetLRByKeyResponseDocument
getLRByKeyRequest(com.metasolv.mip.lsr9API.GetLRByKeyRequestDocument x0);


com.metasolv.mip.lsr9API.GetDLByKeyResponseDocument
getDLByKeyRequest(com.metasolv.mip.lsr9API.GetDLByKeyRequestDocument x0);


com.metasolv.mip.lsr9API.GetLSRByKeyResponseDocument
getLSRByKeyRequest(com.metasolv.mip.lsr9API.GetLSRByKeyRequestDocument x0);


com.metasolv.mip.lsr9API.GetLSRCMByKeyResponseDocument
getLSRCMByKeyRequest(com.metasolv.mip.lsr9API.GetLSRCMByKeyRequestDocument
x0);


com.metasolv.mip.lsr9API.CreateDSCNByValueResponseDocument
createDSCNByValueRequest(com.metasolv.mip.lsr9API.CreateDSCNByValueRequestDo
cument x0);


com.metasolv.mip.lsr9API.CreateDSREDByValueResponseDocument
createDSREDByValueRequest(com.metasolv.mip.lsr9API.CreateDSREDByValueRequest
Document x0);


com.metasolv.mip.lsr9API.CreateLRByValueResponseDocument
createLRByValueRequest(com.metasolv.mip.lsr9API.CreateLRByValueRequestDocume
nt x0);


com.metasolv.mip.lsr9API.CreateLSRCMByValueResponseDocument
createLSRCMByValueRequest(com.metasolv.mip.lsr9API.CreateLSRCMByValueRequest
Document x0);


com.metasolv.mip.lsr9API.CreateNPLSRByValueResponseDocument
createNPLSRByValueRequest(com.metasolv.mip.lsr9API.CreateNPLSRByValueRequest
Document x0);
com.metasolv.mip.lsr9API.QueryCCNAResponseDocument
queryCCNARequest(com.metasolv.mip.lsr9API.QueryCCNARequestDocument x0);


com.metasolv.mip.lsr9API.QueryLSRResponseDocument
queryLSRRequest(com.metasolv.mip.lsr9API.QueryLSRRequestDocument x0);


com.metasolv.mip.lsr9API.QueryLSRForPONCCNAVERResponseDocument
queryLSRForPONCCNAVERRequest(com.metasolv.mip.lsr9API.QueryLSRForPONCCNAVERR
equestDocument x0);


com.metasolv.mip.lsr9API.QueryPONSForCCNAResponseDocument
queryPONSForCCNARequest(com.metasolv.mip.lsr9API.QueryPONSForCCNARequestDocu
ment x0);


com.metasolv.mip.lsr9API.CreateLSROrderByValueResponseDocument
createLSROrderByValueRequest(com.metasolv.mip.lsr9API.CreateLSROrderByValueR
equestDocument x0);

}
```

Appendix C: Navigating the XSD

The information in this appendix describes how to navigate through the Request and Response structures defined in the xsd. The Request and Response structures defined in the xsd are used by the control methods as input and output parameters. Several examples will show screen shots of the xsd in various states of expansion. You can view the xsd in such a manner by using a tool such as XMLSpy.

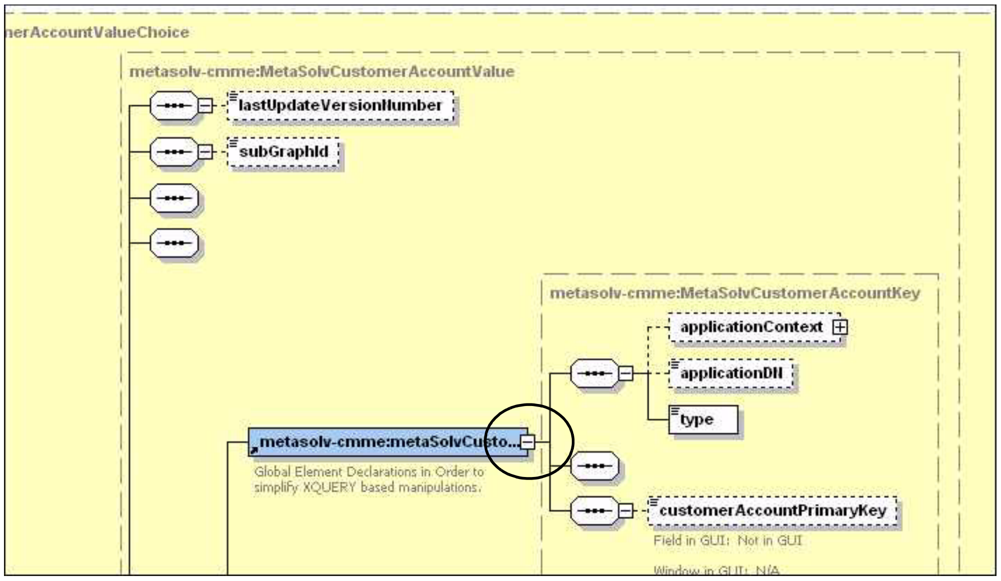
XMLSpy offers several ways to view xml. You may be used to a more traditional view of xml, such as the text view shown below. However, this can become very difficult to read when dealing with large structures because typically elements within the structure reference other structures, which you then have scroll around to locate. Therefore, these examples show how to view the xml using the "Schema/WSDL Design View", which allows you to the view top level structures and then expand and collapse them as needed. Viewing the xml structure in this manner automatically pulls in the referenced structures, so there is no need to scroll around to locate them.



```
    <element name="updateCustomerAccountByValueRequest">
        <annotation>
            <appinfo>MetaSolv Customer Management</appinfo>
            <documentation>Creates a new customer account or modifies an existing customer account.
Response: updateCustomerAccountByValueResponse.
</documentation>
        </annotation>
        <complexType>
            <sequence>
                <element ref="metasolv-cmme:value"/>
            </sequence>
        </complexType>
    </element>
    <element name="updateCustomerAccountByValueResponse">
        <annotation>
            <appinfo>MetaSolv Customer Management</appinfo>
            <documentation>Returns the customer account key corresponding to a customer created or updated in the system.
Response: Customer account key.
</documentation>
        </annotation>
        <complexType>
            <sequence>
                <element ref="metasolv-cmme:key"/>
            </sequence>
        </complexType>
    </element>
```

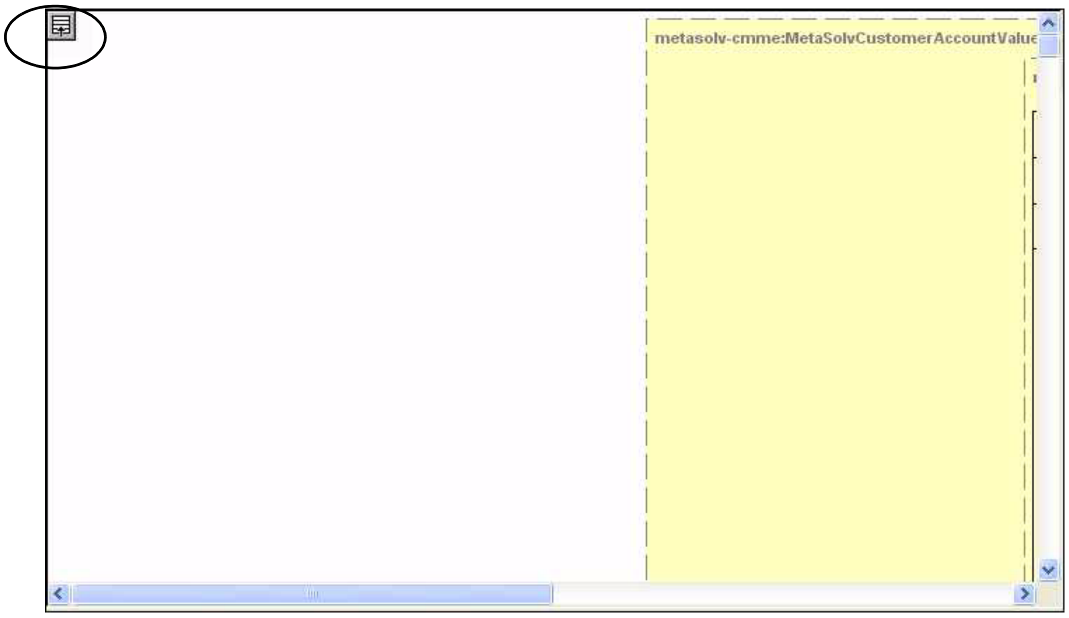
Note:

Before going through the examples, note that the following two screen shots apply to all examples. The examples will explain how to navigate through an xml structure by *expanding* the structure as you read it. If you wish to *collapse* the structures, you can collapase an individual structure by clicking on the "-" as shown below.



nerAccountValueChoice
metasolv-cmme:MetaSolvCustomerAccountValue
lastUpdateVersionNumber
subGraphId
metasolv-cmme:MetaSolvCustomerAccountKey
applicationContext
applicationDN
type
metasolv-cmme:metaSolvCusto...
Global Element Declarations in Order to simplify XQUERY based manipulations.
customerAccountPrimaryKey
Field in GUI: Not in GUI
Window in GUI: N/A

Or, you can collapse the entire structure by clicking on the collapse button as shown below. This button is only visible in the upper left corner, so you must scroll all the way up and all the way to the left to see it.



Example 1: importCustomerAccount

This example shows how a typical XML API import method works. The importCustomerAccount method is defined in the CustomerManagement control class as follows. You can see a full list of controls in Appendix B.

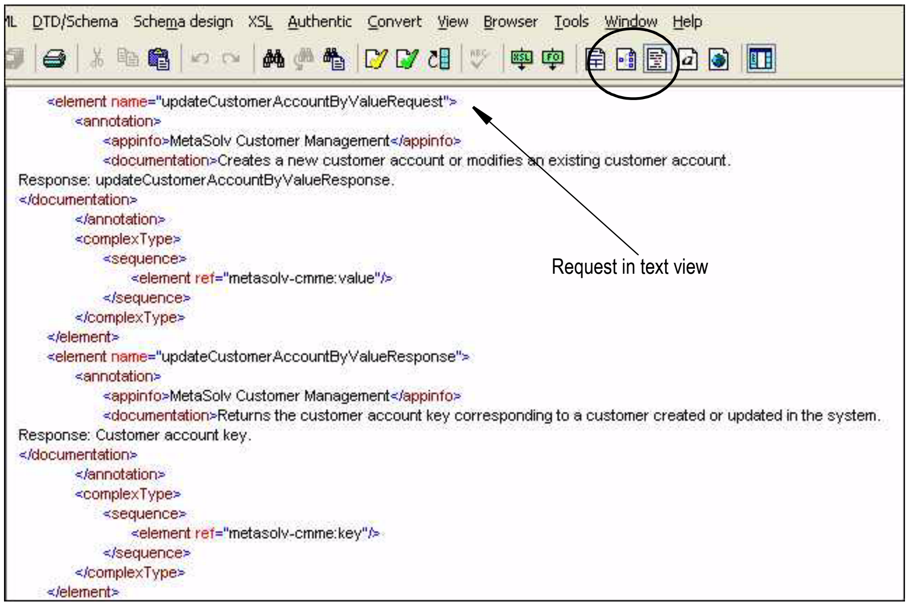
```
com.metasolv.mip.customerManagementAPI.UpdateCustomerAccountByValueRes
ponseDocument
importCustomerAccount(com.metasolv.mip.customerManagementAPI.UpdateCus
tomerAccountByValueRequestDocument request);
```

Request structure

The method defines one input parameter, "request", which is defined as type com.metasolv.mip.customerManagementAPI.UpdateCustomerAccountByValueRequestDocument. This tells us that an xml structure named *updateCustomerAccountByValueRequest* is defined in the CustomerManagementAPI.xsd. Therefore, we will examine the xml structure *updateCustomerAccountByValueRequest*, which is the input to the control method importCustomerAccount.

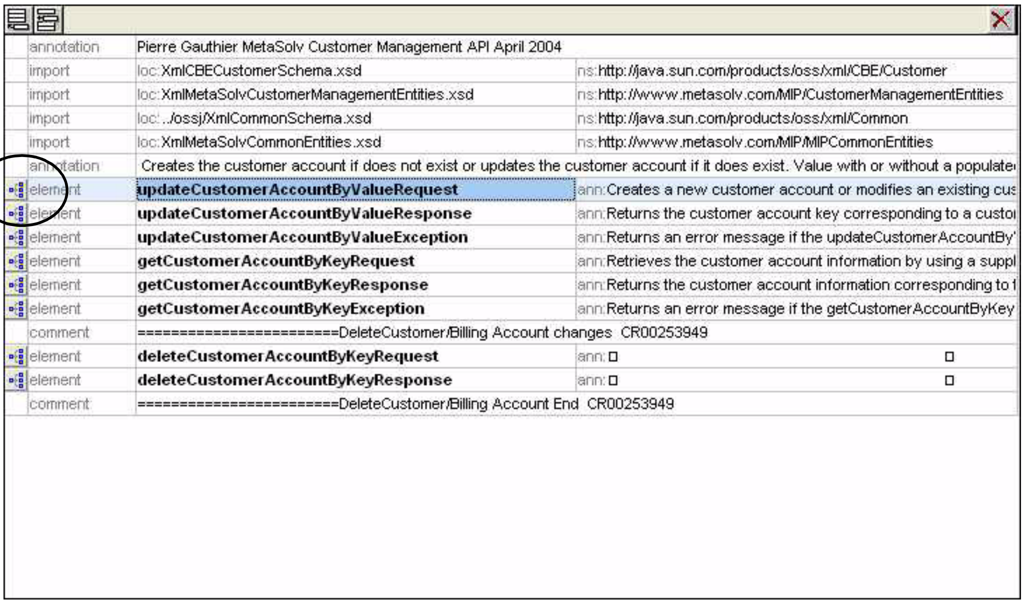
The following steps will walk you through viewing and understanding the UpdateCustomerAccountByValueRequest structure.

1. Open the CustomerManagementAPI.xsd, using an xml tool such as XMLSpy.
2. If the xml file comes up as text view, change the view by selecting a different view as shown below.


L DTD/Schema Schema design XSL Authentic Convert View Browser Tools Window Help
Request in text view

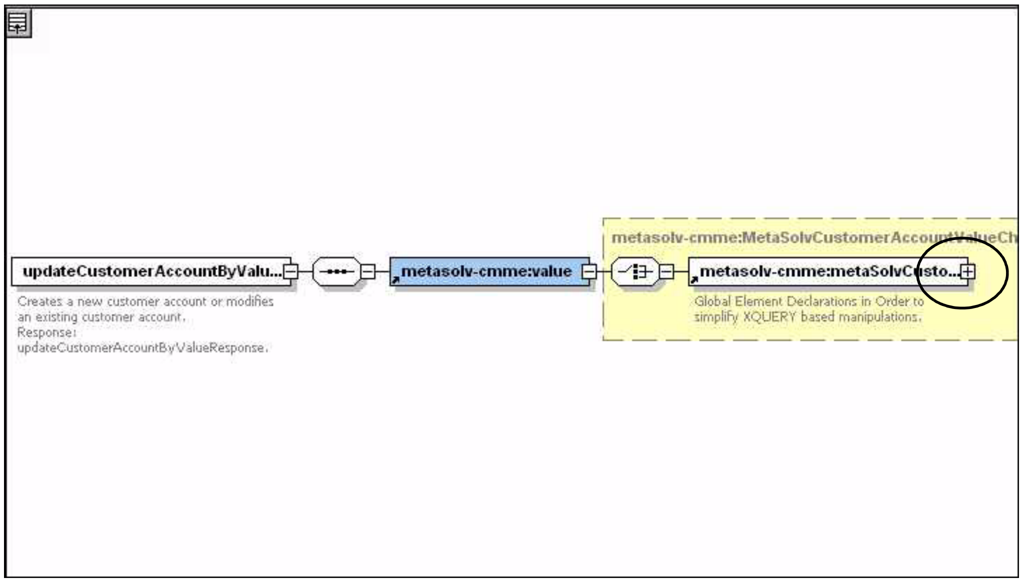
```
    <element name="updateCustomerAccountByValueRequest">
        <annotation>
            <appinfo>MetaSolv Customer Management</appinfo>
            <documentation>Creates a new customer account or modifies an existing customer account.
Response: updateCustomerAccountByValueResponse.
</documentation>
        </annotation>
        <complexType>
            <sequence>
                <element ref="metasolv-cmme:value"/>
            </sequence>
        </complexType>
    </element>
    <element name="updateCustomerAccountByValueResponse">
        <annotation>
            <appinfo>MetaSolv Customer Management</appinfo>
            <documentation>Returns the customer account key corresponding to a customer created or updated in the system.
Response: Customer account key.
</documentation>
        </annotation>
        <complexType>
            <sequence>
                <element ref="metasolv-cmme:key"/>
            </sequence>
        </complexType>
    </element>
```

3. The top-level structures are now clearly listed. Expand the updateCustomerAccountByValueRequest structure by clicking on the expand button as indicated below.

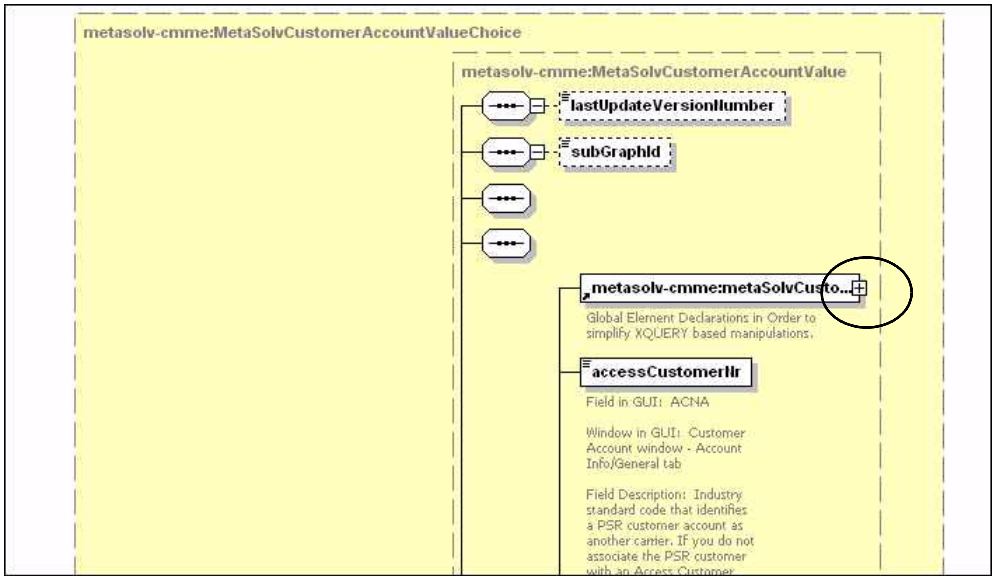


annotation	Pierre Gauthier MetaSolv Customer Management API April 2004	
import	loc: XmlCBECustomerSchema.xsd	ns: http://java.sun.com/products/oss/xml/CBE/Customer
import	loc: XmlMetaSolvCustomerManagementEntities.xsd	ns: http://www.metasolv.com/MIP/CustomerManagementEntities
import	loc: ../ossj/XmlCommonSchema.xsd	ns: http://java.sun.com/products/oss/xml/Common
import	loc: XmlMetaSolvCommonEntities.xsd	ns: http://www.metasolv.com/MIP/MIPCommonEntities
annotation	Creates the customer account if does not exist or updates the customer account if it does exist. Value with or without a populate	
element	**updateCustomerAccountByValueRequest**	ann: Creates a new customer account or modifies an existing cus
element	**updateCustomerAccountByValueResponse**	ann: Returns the customer account key corresponding to a custo
element	**updateCustomerAccountByValueException**	ann: Returns an error message if the updateCustomerAccountBy
element	**getCustomerAccountByKeyRequest**	ann: Retrieves the customer account information by using a suppl
element	**getCustomerAccountByKeyResponse**	ann: Returns the customer account information corresponding to t
element	**getCustomerAccountByKeyException**	ann: Returns an error message if the getCustomerAccountByKey
comment	========================DeleteCustomer/Billing Account changes CR00253949	
element	**deleteCustomerAccountByKeyRequest**	ann: □ □
element	**deleteCustomerAccountByKeyResponse**	ann: □ □
comment	========================DeleteCustomer/Billing Account End CR00253949	

4. You are now viewing the contents of the updateCustomerAccountByValueReqeust structure. Further expand the udpateCustomerAccountByValueRequest structure by clicking on the expand button as indicated below.


updateCustomerAccountByValu...
Creates a new customer account or modifies an existing customer account. Response: updateCustomerAccountByValueResponse.
metasolv-cmme:value
metasolv-cmme:MetaSolvCustomerAccountValueCh
metasolv-cmme:metaSolvCusto...
Global Element Declarations in Order to simplify XQUERY based manipulations.

5. You are now viewing elements that define a data type. These elements will house the data that comprise the customer account being imported. For example, lastUpdateVersionNumber and subGraphId. The metaSolvCustomerAccountKey defines another structure. Click the "+" to further expand the structure.


metasolv-cmme:MetaSolvCustomerAccountValueChoice
metasolv-cmme:MetaSolvCustomerAccountValue
lastUpdateVersionNumber
subGraphId
metasolv-cmme:metaSolvCusto...
accessCustomerNr

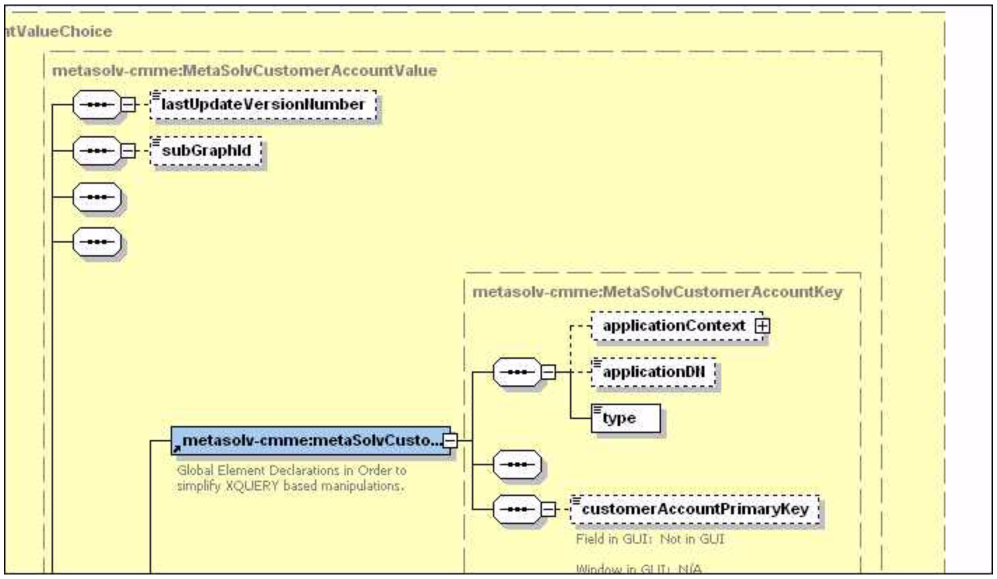
Global Element Declarations in Order to simplify XQUERY based manipulations.

Field in GUI: ACNA

Window in GUI: Customer Account window - Account Info/General tab

Field Description: Industry standard code that identifies a PSR customer account as another carrier. If you do not associate the PSR customer with an Access Customer

6. You are now viewing additional elements that define a data type, such as applicationDN, type, and customerAccountPrimaryKey.

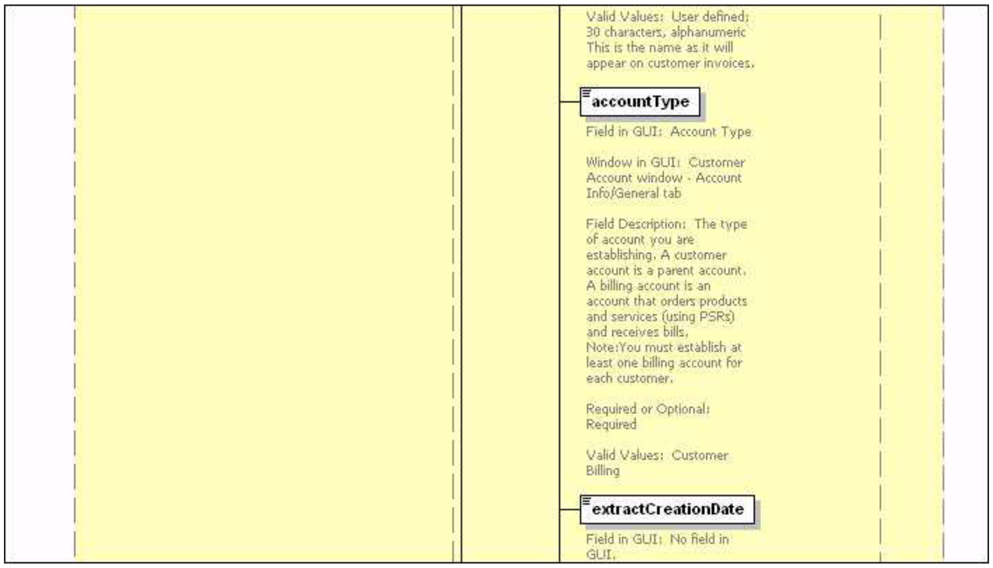

tValueChoice
metasolv-cmme:MetaSolvCustomerAccountValue
lastUpdateVersionNumber
subGraphId
metasolv-cmme:MetaSolvCustomerAccountKey
applicationContext
applicationDN
type
metasolv-cmme:metaSolvCusto...
customerAccountPrimaryKey

Global Element Declarations in Order to simplify XQUERY based manipulations.

Field in GUI: Not in GUI

Window in GUI: N/A

7. Scroll down, and you can view field properties, such as Field in GUI, Window in GUI, Field Description, Required or Optional, and Valid Values.



Valid Values: User defined; 30 characters, alphanumeric This is the name as it will appear on customer invoices.

accountType

Field in GUI: Account Type

Window in GUI: Customer Account window - Account Info/General tab

Field Description: The type of account you are establishing. A customer account is a parent account. A billing account is an account that orders products and services (using PSRs) and receives bills. Note:You must establish at least one billing account for each customer.

Required or Optional: Required

Valid Values: Customer Billing

extractCreationDate

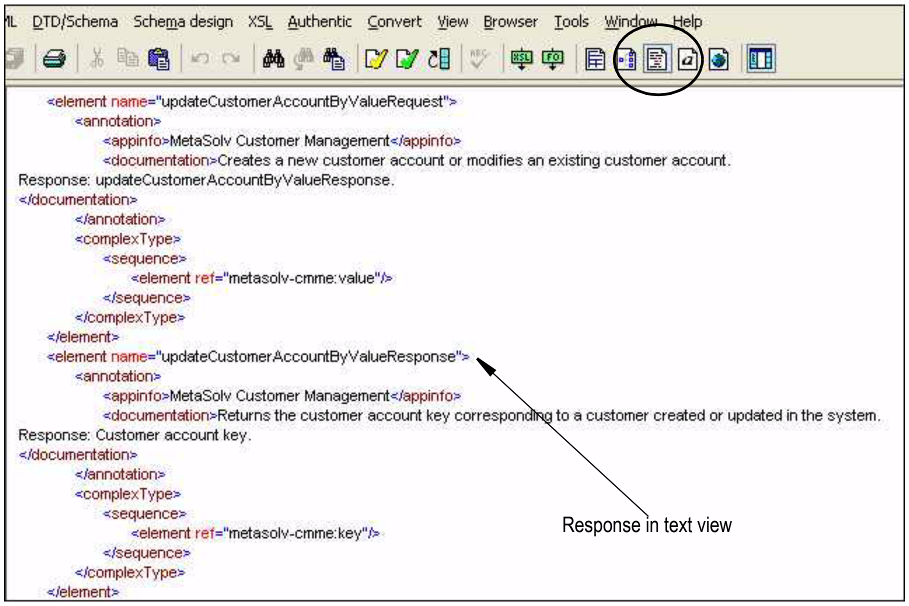
Field in GUI: No field in GUI.

Response structure

The method defines it's return as type com.metasolv.mip.customerManagementAPI.UpdateCustomerAccountByValueResponseDocument. This tells us that an xml structure named *updateCustomerAccountByValueResponse* is defined in the CustomerManagementAPI.xsd. Therefore, we will examine the xml structure *updateCustomerAccountByValueResponse*, which is what is returned by the control method importCustomerAccount.

The following steps will walk you through viewing and understanding the UpdateCustomerAccountByValueResponse structure.

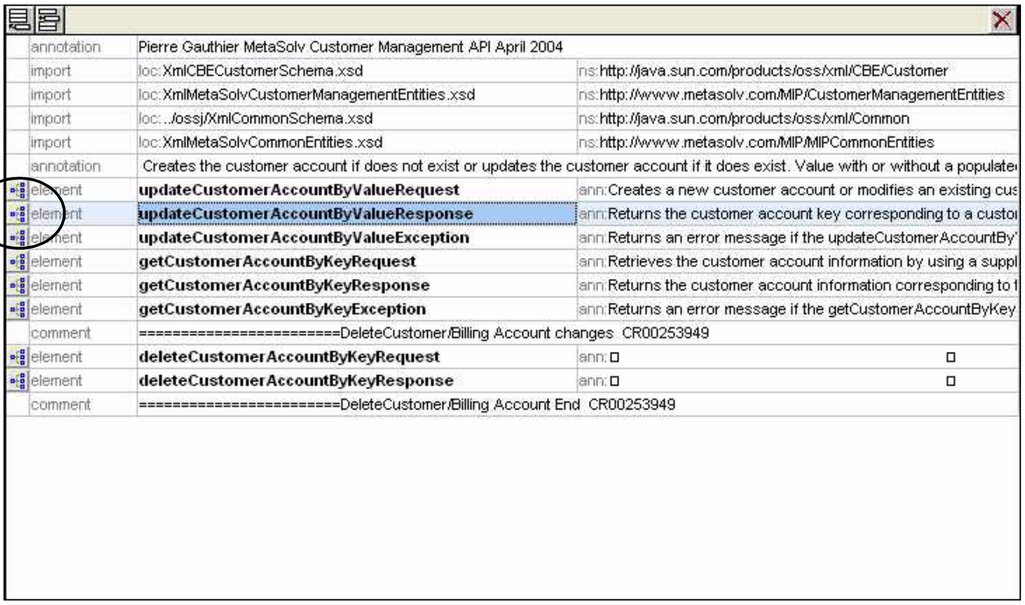
1. Open the CustomerManagementAPI.xsd, using an xml tool such as XMLSpy.
2. If the xml file comes up as text view, change the view by selecting a different view as shown below.



ML DTD/Schema Schema design XSL Authentic Convert View Browser Tools Window Help
Response in text view

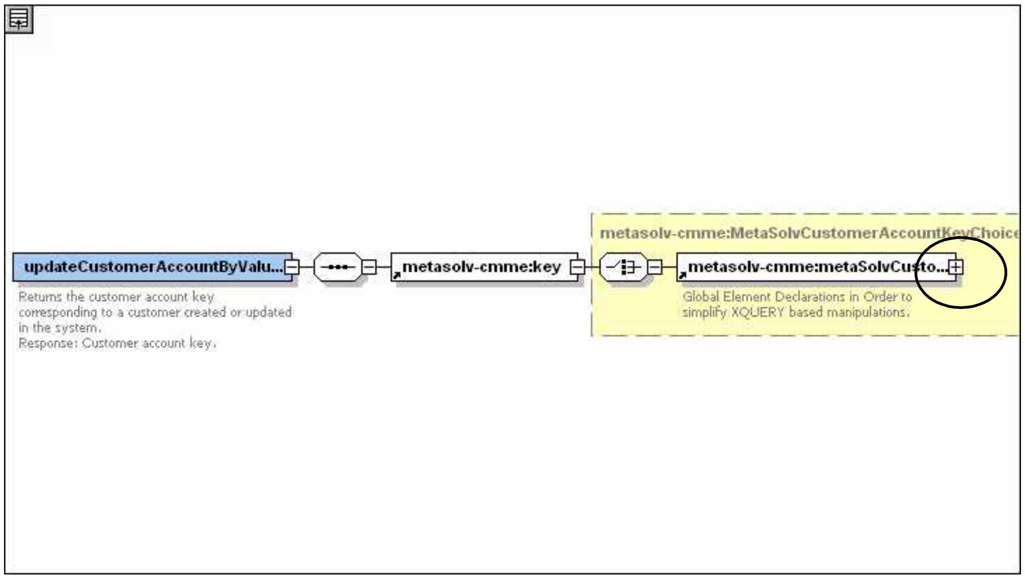
```
    <element name="updateCustomerAccountByValueRequest">
        <annotation>
            <appinfo>MetaSolv Customer Management</appinfo>
            <documentation>Creates a new customer account or modifies an existing customer account.
Response: updateCustomerAccountByValueResponse.
</documentation>
        </annotation>
        <complexType>
            <sequence>
                <element ref="metasolv-cmme:value"/>
            </sequence>
        </complexType>
    </element>
    <element name="updateCustomerAccountByValueResponse">
        <annotation>
            <appinfo>MetaSolv Customer Management</appinfo>
            <documentation>Returns the customer account key corresponding to a customer created or updated in the system.
Response: Customer account key.
</documentation>
        </annotation>
        <complexType>
            <sequence>
                <element ref="metasolv-cmme:key"/>
            </sequence>
        </complexType>
    </element>
```

3. The top-level structures are now clearly listed. Expand the updateCustomerAccountByValueResponse structure by clicking on the expand button as indicated below.



annotation	Pierre Gauthier MetaSolv Customer Management API April 2004	
import	loc: XmlCBECustomerSchema.xsd	ns: http://java.sun.com/products/oss/xml/CBE/Customer
import	loc: XmlMetaSolvCustomerManagementEntities.xsd	ns: http://www.metasolv.com/MIP/CustomerManagementEntities
import	loc: ../ossj/XmlCommonSchema.xsd	ns: http://java.sun.com/products/oss/xml/Common
import	loc: XmlMetaSolvCommonEntities.xsd	ns: http://www.metasolv.com/MIP/MIPCommonEntities
annotation	Creates the customer account if does not exist or updates the customer account if it does exist. Value with or without a populate	
element	**updateCustomerAccountByValueRequest**	ann: Creates a new customer account or modifies an existing cus
element	**updateCustomerAccountByValueResponse**	ann: Returns the customer account key corresponding to a custo
element	**updateCustomerAccountByValueException**	ann: Returns an error message if the updateCustomerAccountBy
element	**getCustomerAccountByKeyRequest**	ann: Retrieves the customer account information by using a suppl
element	**getCustomerAccountByKeyResponse**	ann: Returns the customer account information corresponding to t
element	**getCustomerAccountByKeyException**	ann: Returns an error message if the getCustomerAccountByKey
comment	=======================DeleteCustomer/Billing Account changes CR00253949	
element	**deleteCustomerAccountByKeyRequest**	ann: □ □
element	**deleteCustomerAccountByKeyResponse**	ann: □ □
comment	=======================DeleteCustomer/Billing Account End CR00253949	

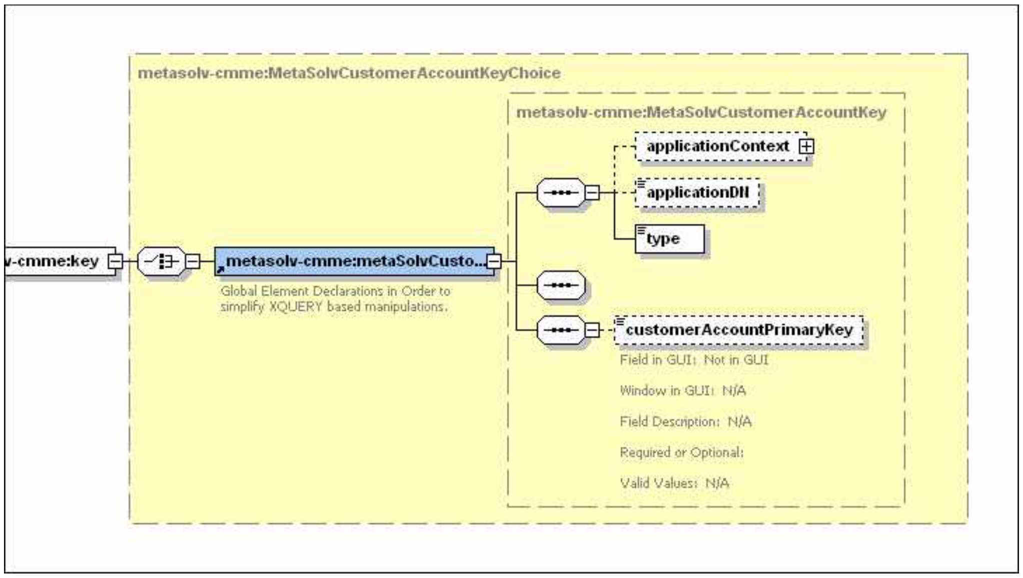
4. You are now viewing the contents of the updateCustomerAccountByValueResponse structure. Further expand the udpateCustomerAccountByValueResponse structure by clicking on the expand button as indicated below.


updateCustomerAccountByValu...
metasolv-cmme:key
metasolv-cmme:MetaSolvCustomerAccountKeyChoice
metasolv-cmme:metaSolvCusto...

Returns the customer account key corresponding to a customer created or updated in the system.
Response: Customer account key.

Global Element Declarations in Order to simplify XQUERY based manipulations.

5. You are now viewing elements that define a data type. These elements will house the data that is returned in the response regarding the customer account that was imported. For example, applicationDN, type, and customerAccountPrimaryKey.


metasolv-cmme:MetaSolvCustomerAccountKeyChoice
metasolv-cmme:MetaSolvCustomerAccountKey
applicationContext
applicationDN
type
v-cmme:key
metasolv-cmme:metaSolvCusto...
Global Element Declarations in Order to simplify XQUERY based manipulations.
customerAccountPrimaryKey
Field in GUI: Not in GUI
Window in GUI: N/A
Field Description: N/A
Required or Optional:
Valid Values: N/A

Example 2: getCustomerAccountByKey

This example shows how a typical XML API export method works.The getCustomerAccountByKey method is defined in the CustomerManagement control class as follows. You can see a full list of controls in Appendix B.

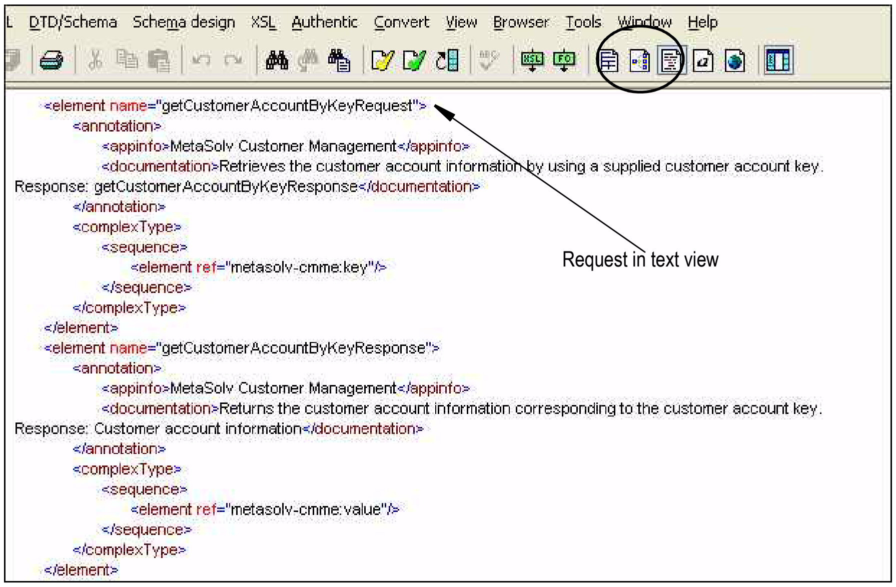
```
com.metasolv.mip.customerManagementAPI.GetCustomerAccountByKeyResponse
Document
getCustomerAccountByKey(com.metasolv.mip.customerManagementAPI.GetCust
omerAccountByKeyRequestDocument request);
```

Request structure

The method defines one input parameter, "request", which is defined as type com.metasolv.mip.customerManagementAPI.GetCustomerAccountByKeyRequestDocument. This tells us that an xml structure named *getCustomerAccountByKeyRequest* is defined in the CustomerManagementAPI.xsd. Therefore, we will examine the xml structure *getCustomerAccountByKeyRequest*, which is the input to the control method getCustomerAccountByKey.

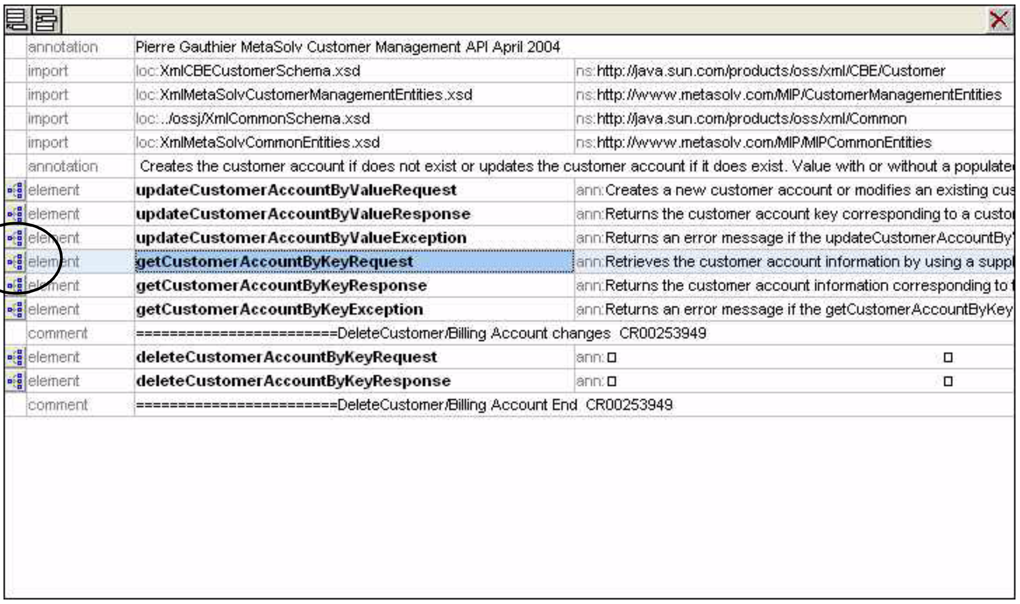
The following steps will walk you through viewing and understanding the GetCustomerAccountByKeyRequest structure.

1. Open the CustomerManagementAPI.xsd, using an xml tool such as XMLSpy.
2. If the xml file comes up as text view, change the view by selecting a different view as shown below.


L DTD/Schema Schema design XSL Authentic Convert View Browser Tools Window Help
Request in text view

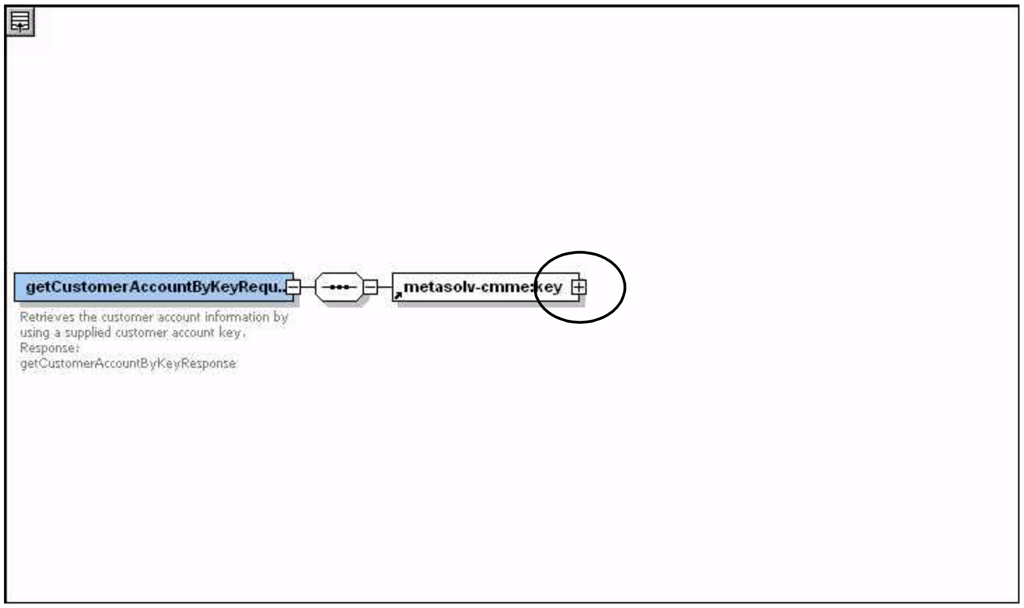
```
    <element name="getCustomerAccountByKeyRequest">
        <annotation>
            <appinfo>MetaSolv Customer Management</appinfo>
            <documentation>Retrieves the customer account information by using a supplied customer account key.
Response: getCustomerAccountByKeyResponse</documentation>
        </annotation>
        <complexType>
            <sequence>
                <element ref="metasolv-cmme:key"/>
            </sequence>
        </complexType>
    </element>
    <element name="getCustomerAccountByKeyResponse">
        <annotation>
            <appinfo>MetaSolv Customer Management</appinfo>
            <documentation>Returns the customer account information corresponding to the customer account key.
Response: Customer account information</documentation>
        </annotation>
        <complexType>
            <sequence>
                <element ref="metasolv-cmme:value"/>
            </sequence>
        </complexType>
    </element>
```

3. The top-level structures are now clearly listed. Expand the getCustomerAccountByKeyRequest structure by clicking on the expand button as indicated below.



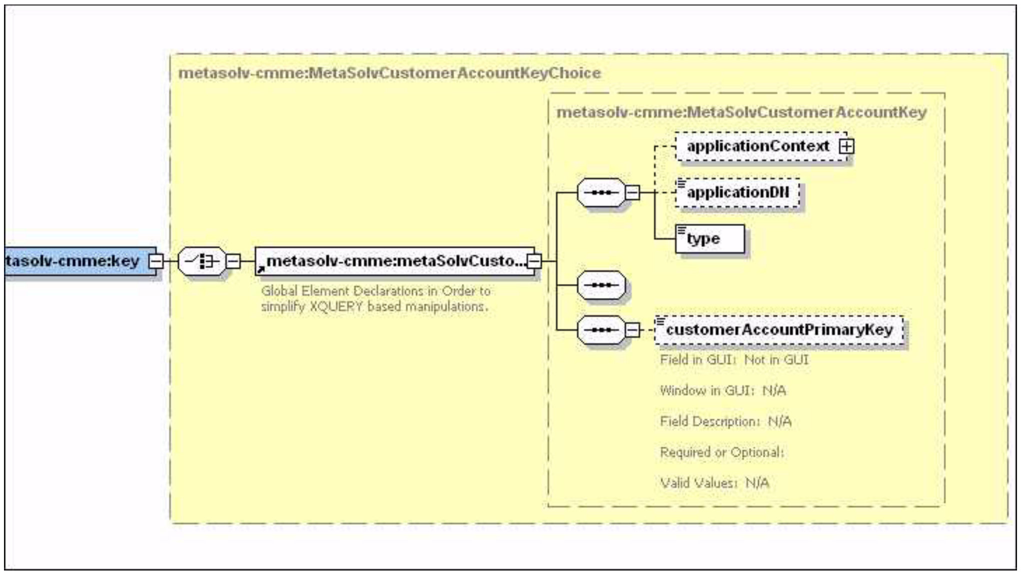
annotation	Pierre Gauthier MetaSolv Customer Management API April 2004	
import	loc: XmlCBECustomerSchema.xsd	ns: http://java.sun.com/products/oss/xml/CBE/Customer
import	loc: XmlMetaSolvCustomerManagementEntities.xsd	ns: http://www.metasolv.com/MIP/CustomerManagementEntities
import	loc: ../ossj/XmlCommonSchema.xsd	ns: http://java.sun.com/products/oss/xml/Common
import	loc: XmlMetaSolvCommonEntities.xsd	ns: http://www.metasolv.com/MIP/MIPCommonEntities
annotation	Creates the customer account if does not exist or updates the customer account if it does exist. Value with or without a populate	
element	**updateCustomerAccountByValueRequest**	ann: Creates a new customer account or modifies an existing cus
element	**updateCustomerAccountByValueResponse**	ann: Returns the customer account key corresponding to a custo
element	**updateCustomerAccountByValueException**	ann: Returns an error message if the updateCustomerAccountBy
element	**getCustomerAccountByKeyRequest**	ann: Retrieves the customer account information by using a suppl
element	**getCustomerAccountByKeyResponse**	ann: Returns the customer account information corresponding to
element	**getCustomerAccountByKeyException**	ann: Returns an error message if the getCustomerAccountByKey
comment	========================DeleteCustomer/Billing Account changes CR00253949	
element	**deleteCustomerAccountByKeyRequest**	ann: ☐ ☐
element	**deleteCustomerAccountByKeyResponse**	ann: ☐ ☐
comment	========================DeleteCustomer/Billing Account End CR00253949	

4. You are now viewing the contents of the getCustomerAccountByKeyReqeust structure. Further expand the getCustomerAccountByKeyRequest structure by clicking on the expand button as indicated below.



Retrieves the customer account information by using a supplied customer account key.
Response:
getCustomerAccountByKeyResponse

5. You are now viewing elements that define a data type. These elements will house the data that is required to get the customer account. For example, applicationDN, type, and customerAccountPrimaryKey.

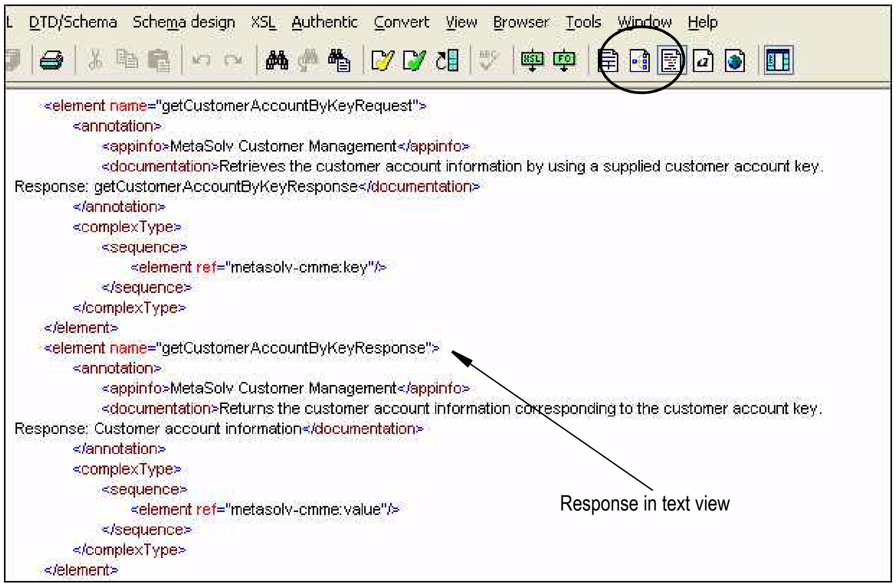

metasolv-cmme:MetaSolvCustomerAccountKeyChoice
metasolv-cmme:MetaSolvCustomerAccountKey
applicationContext
applicationDN
type
tasolv-cmme:key
metasolv-cmme:metaSolvCusto...
Global Element Declarations in Order to simplify XQUERY based manipulations.
customerAccountPrimaryKey
Field in GUI: Not in GUI
Window in GUI: N/A
Field Description: N/A
Required or Optional:
Valid Values: N/A

Response structure

The method defines it's return as type com.metasolv.mip.customerManagementAPI.GetCustomerAccountByKeyResponseDocument. This tells us that an xml structure named *getCustomerAccountByKeyResponse* is defined in the CustomerManagementAPI.xsd. Therefore, we will examine the xml structure *getCustomerAccountByKeyResponse*, which is what is returned by the control method getCustomerAccountByKey.

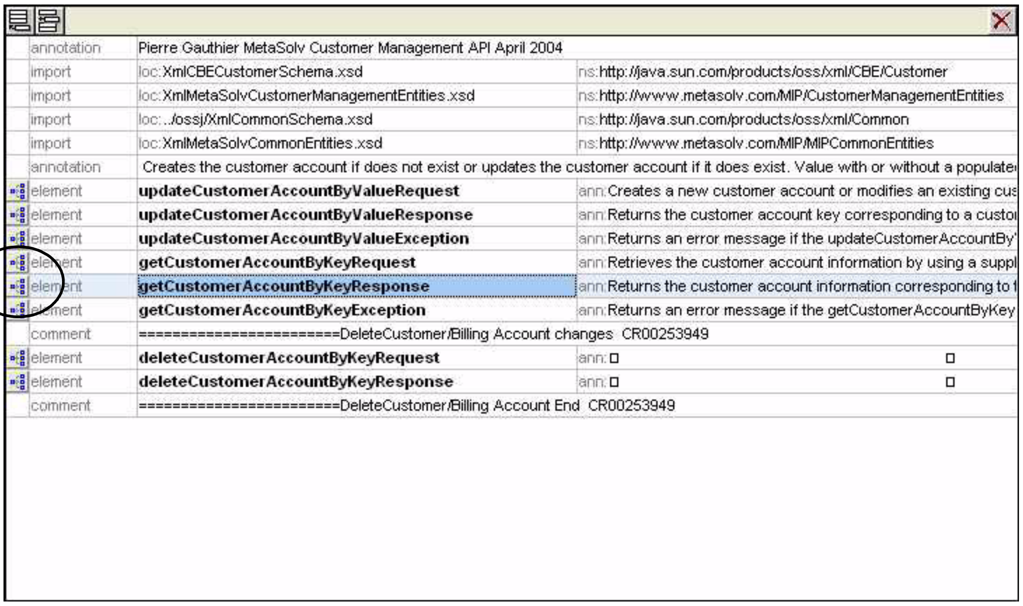
The following steps will walk you through viewing and understanding the UpdateCustomerAccountByValueRequest structure.

1. Open the CustomerManagementAPI.xsd, using an xml tool such as XMLSpy.
2. If the xml file comes up as text view, change the view by selecting a different view as shown below.


L DTD/Schema Schema design XSL Authentic Convert View Browser Tools Window Help
Response in text view

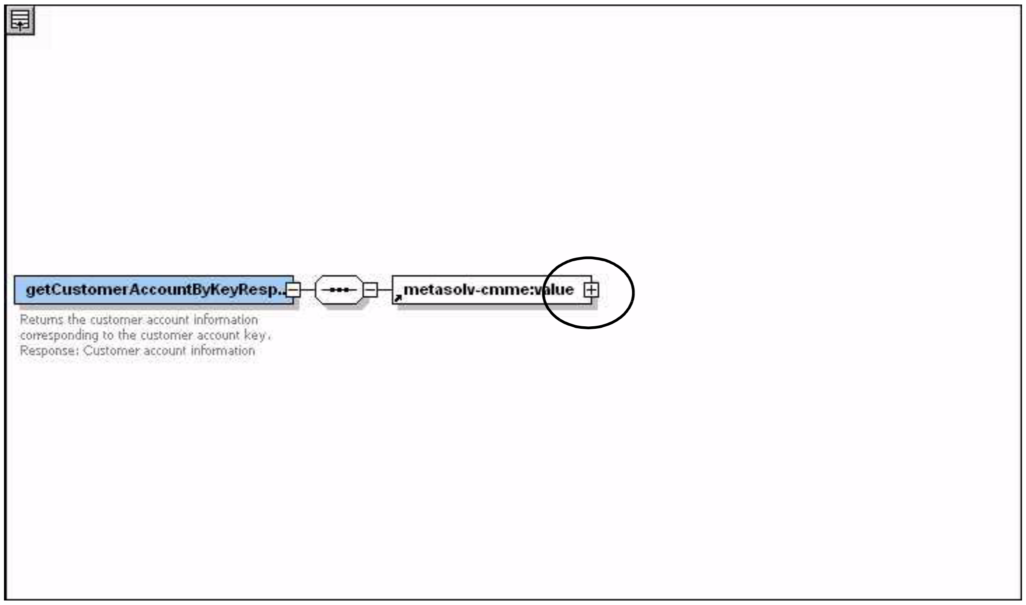
```
    <element name="getCustomerAccountByKeyRequest">
        <annotation>
            <appinfo>MetaSolv Customer Management</appinfo>
            <documentation>Retrieves the customer account information by using a supplied customer account key.
Response: getCustomerAccountByKeyResponse</documentation>
        </annotation>
        <complexType>
            <sequence>
                <element ref="metasolv-cmme:key"/>
            </sequence>
        </complexType>
    </element>
    <element name="getCustomerAccountByKeyResponse">
        <annotation>
            <appinfo>MetaSolv Customer Management</appinfo>
            <documentation>Returns the customer account information corresponding to the customer account key.
Response: Customer account information</documentation>
        </annotation>
        <complexType>
            <sequence>
                <element ref="metasolv-cmme:value"/>
            </sequence>
        </complexType>
    </element>
```

3. The top-level structures are now clearly listed. Expand the getCustomerAccountByKeyResponse structure by clicking on the expand button as indicated below.



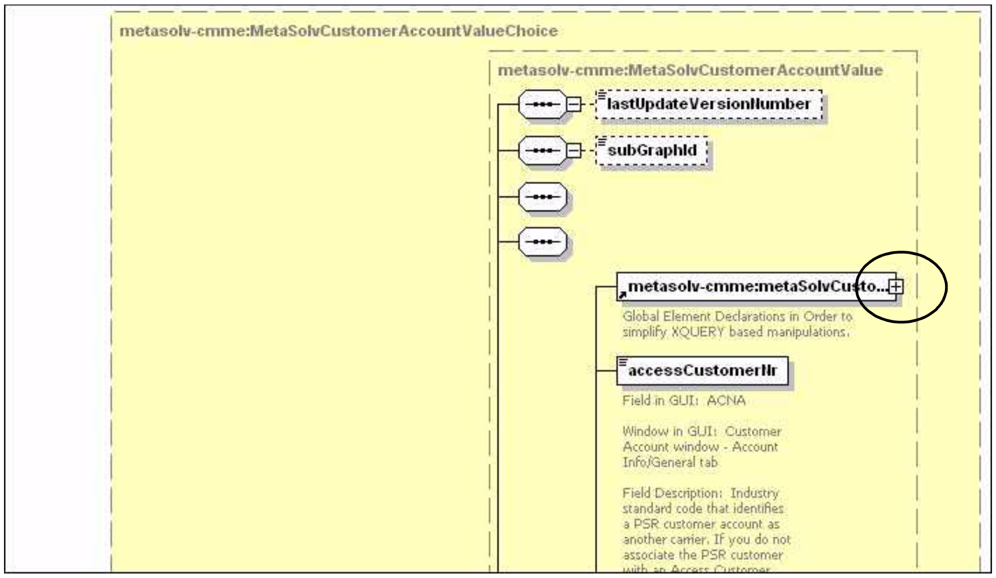
annotation	Pierre Gauthier MetaSolv Customer Management API April 2004	
import	loc:XmlCBECustomerSchema.xsd	ns:http://java.sun.com/products/oss/xml/CBE/Customer
import	loc:XmlMetaSolvCustomerManagementEntities.xsd	ns:http://www.metasolv.com/MIP/CustomerManagementEntities
import	loc:../ossj/XmlCommonSchema.xsd	ns:http://java.sun.com/products/oss/xml/Common
import	loc:XmlMetaSolvCommonEntities.xsd	ns:http://www.metasolv.com/MIP/MIPCommonEntities
annotation	Creates the customer account if does not exist or updates the customer account if it does exist. Value with or without a populate	
element	**updateCustomerAccountByValueRequest**	ann:Creates a new customer account or modifies an existing cus
element	**updateCustomerAccountByValueResponse**	ann:Returns the customer account key corresponding to a custo
element	**updateCustomerAccountByValueException**	ann:Returns an error message if the updateCustomerAccountBy
element	**getCustomerAccountByKeyRequest**	ann:Retrieves the customer account information by using a suppl
element	**getCustomerAccountByKeyResponse**	ann:Returns the customer account information corresponding to t
element	**getCustomerAccountByKeyException**	ann:Returns an error message if the getCustomerAccountByKey
comment	=======================DeleteCustomer/Billing Account changes CR00253949	
element	**deleteCustomerAccountByKeyRequest**	ann:
element	**deleteCustomerAccountByKeyResponse**	ann:
comment	=======================DeleteCustomer/Billing Account End CR00253949	

4. You are now viewing the contents of the getCustomerAccountByKeyResponse structure. Further expand the getCustomerAccountByKeyResponse structure by clicking on the expand button as indicated below.



Returns the customer account information corresponding to the customer account key. Response: Customer account information

5. You are now viewing elements that define a data type. These elements will house the data that is returned in the response regarding the customer account being exported. For example, lastUpdateVersionNumber, subGraphId, and accessCustomerNr. The metaSolvCustomerAccountKey defines another structure. Click the "+" to further expand the structure.


metasolv-cmme:MetaSolvCustomerAccountValueChoice
metasolv-cmme:MetaSolvCustomerAccountValue
lastUpdateVersionNumber
subGraphId
metasolv-cmme:metaSolvCusto...
accessCustomerNr

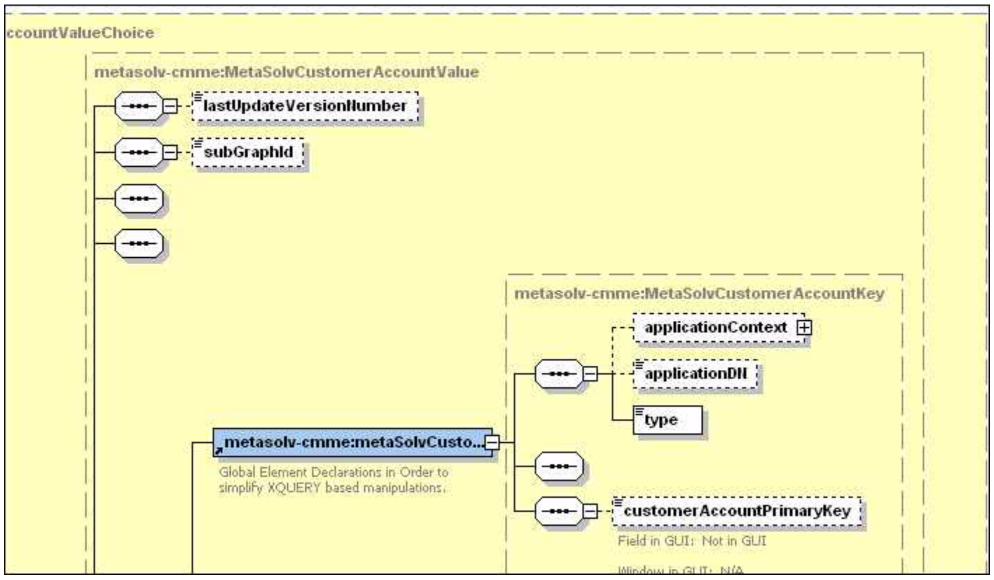
Global Element Declarations in Order to simplify XQUERY based manipulations.

Field in GUI: ACNA

Window in GUI: Customer Account window - Account Info/General tab

Field Description: Industry standard code that identifies a PSR customer account as another carrier. If you do not associate the PSR customer

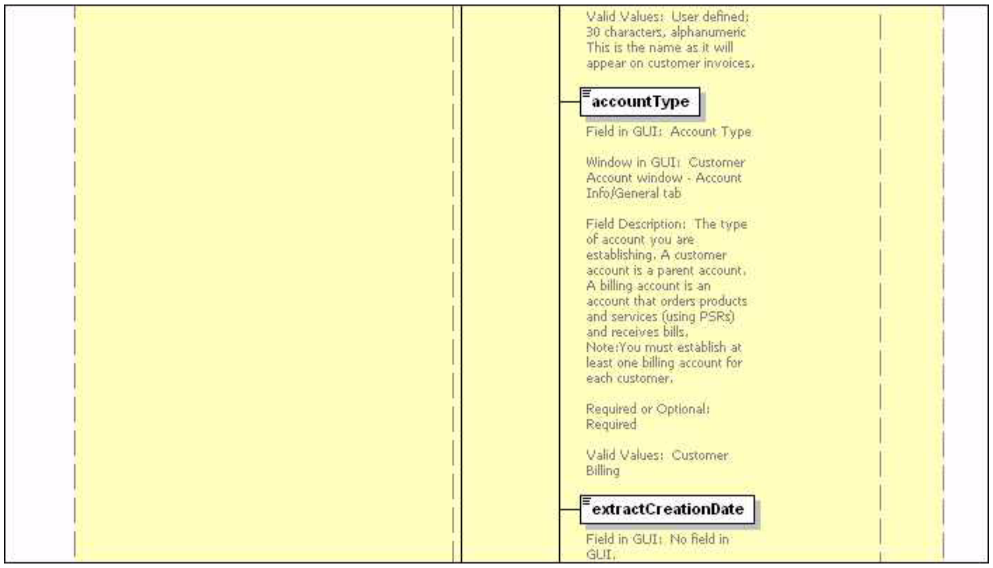
6. You are now viewing additional elements that define a data type, such as applicationDN, type, and customerAccountPrimaryKey.


ccountValueChoice
metasolv-cmme:MetaSolvCustomerAccountValue
lastUpdateVersionNumber
subGraphId
metasolv-cmme:MetaSolvCustomerAccountKey
applicationContext
applicationDN
type
metasolv-cmme:metaSolvCusto...
customerAccountPrimaryKey

Global Element Declarations in Order to simplify XQUERY based manipulations.

Field in GUI: Not in GUI

7. Scroll down, and you can view field properties, such as Field in GUI, Window in GUI, Field Description, Required or Optional, and Valid Values.



Valid Values: User defined; 30 characters, alphanumeric This is the name as it will appear on customer invoices.

accountType

Field in GUI: Account Type

Window in GUI: Customer Account window - Account Info/General tab

Field Description: The type of account you are establishing. A customer account is a parent account. A billing account is an account that orders products and services (using PSRs) and receives bills. Note:You must establish at least one billing account for each customer.

Required or Optional: Required

Valid Values: Customer Billing

extractCreationDate

Field in GUI: No field in GUI.

Example 3: createEntityByValueRequest

This example shows how a typical XML API method that defines choices of structures within the Request and Response structures works.The createEntityByValueRequest method is defined in the InventoryManagement control class as follows. You can see a full list of controls in Appendix B.

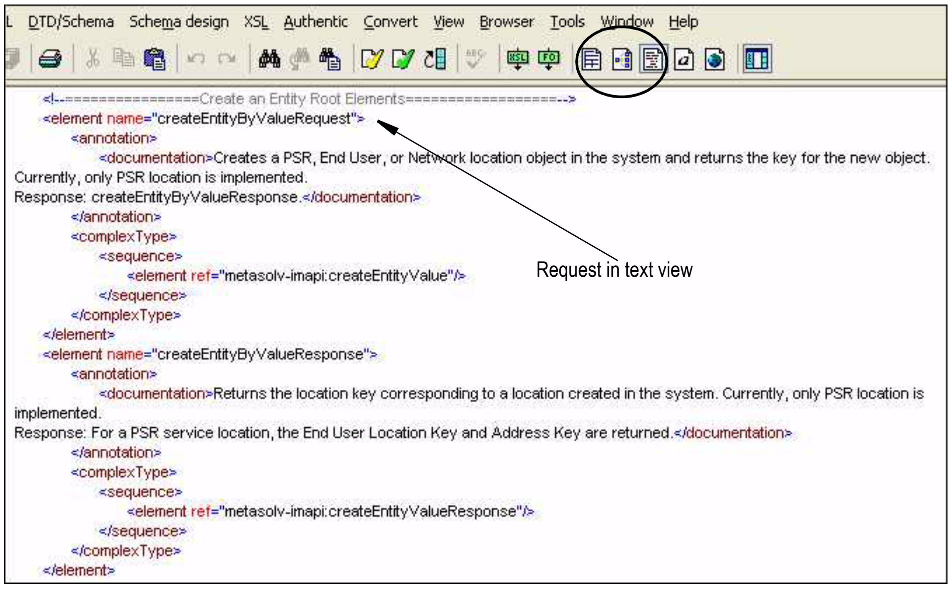
```
com.metasolv.mip.inventoryManagementAPI.CreateEntityByValueResponseDoc
ument
createEntityByValueRequest(com.metasolv.mip.inventoryManagementAPI.Cre
ateEntityByValueRequestDocument request);
```

Request structure

The method defines one input parameter, "request", which is defined as type com.metasolv.mip.inventoryManagementAPI.CreateEntityByValueRequestDocument. This tells us that an xml structure named *createEntityByValueRequest* is defined in the InventoryManagementAPI.xsd. Therefore, we will examine the xml structure *createEntityByValueRequest*, which is the input to the control method createEntityByValueRequest.

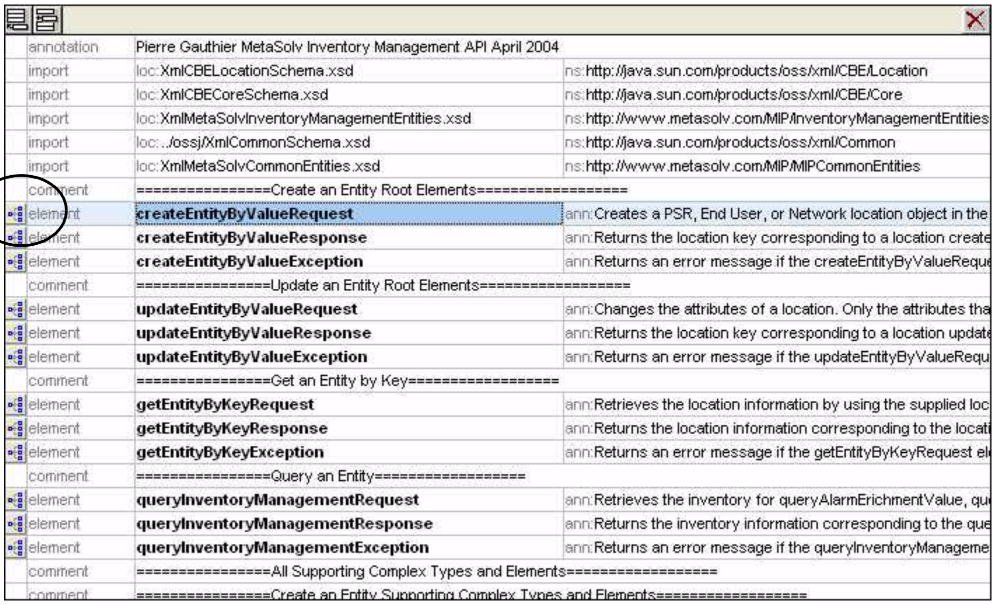
The following steps will walk you through viewing and understanding the CreateEntityByValueRequest structure.

1. Open the InventoryManagementAPI.xsd, using an xml tool such as XMLSpy.
2. If the xml file comes up as text view, change the view by selecting a different view as shown below.


L DTD/Schema Schema design XSL Authentic Convert View Browser Tools Window Help
Request in text view

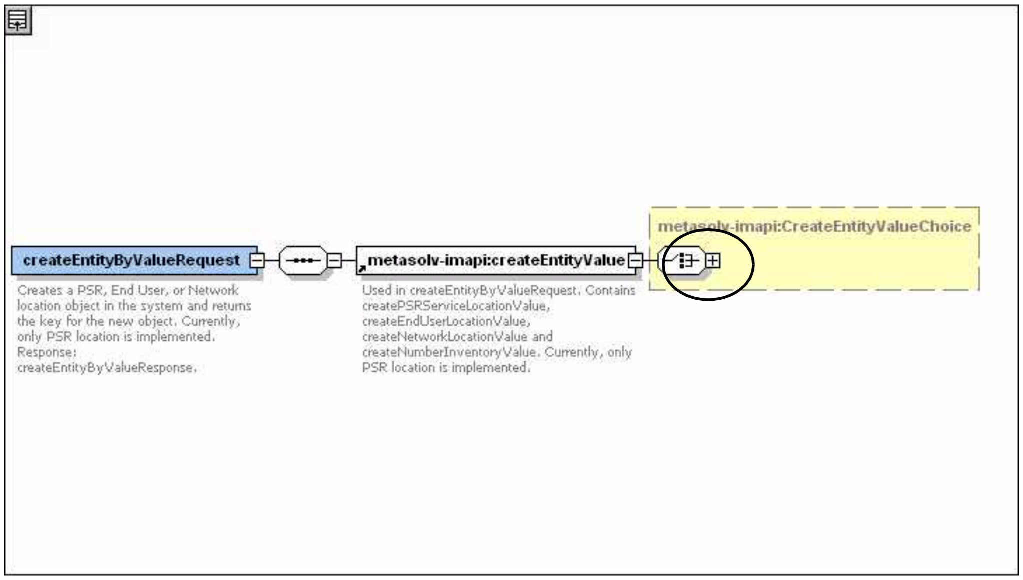
```
    <!--================Create an Entity Root Elements==================-->
    <element name="createEntityByValueRequest">
        <annotation>
            <documentation>Creates a PSR, End User, or Network location object in the system and returns the key for the new object.
Currently, only PSR location is implemented.
Response: createEntityByValueResponse.</documentation>
        </annotation>
        <complexType>
            <sequence>
                <element ref="metasolv-imapi:createEntityValue"/>
            </sequence>
        </complexType>
    </element>
    <element name="createEntityByValueResponse">
        <annotation>
            <documentation>Returns the location key corresponding to a location created in the system. Currently, only PSR location is
implemented.
Response: For a PSR service location, the End User Location Key and Address Key are returned.</documentation>
        </annotation>
        <complexType>
            <sequence>
                <element ref="metasolv-imapi:createEntityValueResponse"/>
            </sequence>
        </complexType>
    </element>
```

3. The top-level structures are now clearly listed. Expand the createEntityByValueRequest structure by clicking on the expand button as indicated below.



annotation	Pierre Gauthier MetaSolv Inventory Management API April 2004	
import	loc: XmlCBELocationSchema.xsd	ns: http://java.sun.com/products/oss/xml/CBE/Location
import	loc: XmlCBECoreSchema.xsd	ns: http://java.sun.com/products/oss/xml/CBE/Core
import	loc: XmlMetaSolvInventoryManagementEntities.xsd	ns: http://www.metasolv.com/MIP/InventoryManagementEntities
import	loc: ../ossj/XmlCommonSchema.xsd	ns: http://java.sun.com/products/oss/xml/Common
import	loc: XmlMetaSolvCommonEntities.xsd	ns: http://www.metasolv.com/MIP/MIPCommonEntities
comment	================Create an Entity Root Elements==================	
element	**createEntityByValueRequest**	ann: Creates a PSR, End User, or Network location object in the
element	**createEntityByValueResponse**	ann: Returns the location key corresponding to a location create
element	**createEntityByValueException**	ann: Returns an error message if the createEntityByValueReque
comment	================Update an Entity Root Elements==================	
element	**updateEntityByValueRequest**	ann: Changes the attributes of a location. Only the attributes tha
element	**updateEntityByValueResponse**	ann: Returns the location key corresponding to a location update
element	**updateEntityByValueException**	ann: Returns an error message if the updateEntityByValueRequ
comment	================Get an Entity by Key==================	
element	**getEntityByKeyRequest**	ann: Retrieves the location information by using the supplied loc
element	**getEntityByKeyResponse**	ann: Returns the location information corresponding to the locati
element	**getEntityByKeyException**	ann: Returns an error message if the getEntityByKeyRequest el
comment	================Query an Entity==================	
element	**queryInventoryManagementRequest**	ann: Retrieves the inventory for queryAlarmErichmentValue, qu
element	**queryInventoryManagementResponse**	ann: Returns the inventory information corresponding to the que
element	**queryInventoryManagementException**	ann: Returns an error message if the queryInventoryManageme
comment	================All Supporting Complex Types and Elements==================	
comment	================Create an Entity Supporting Complex Types and Elements==================	

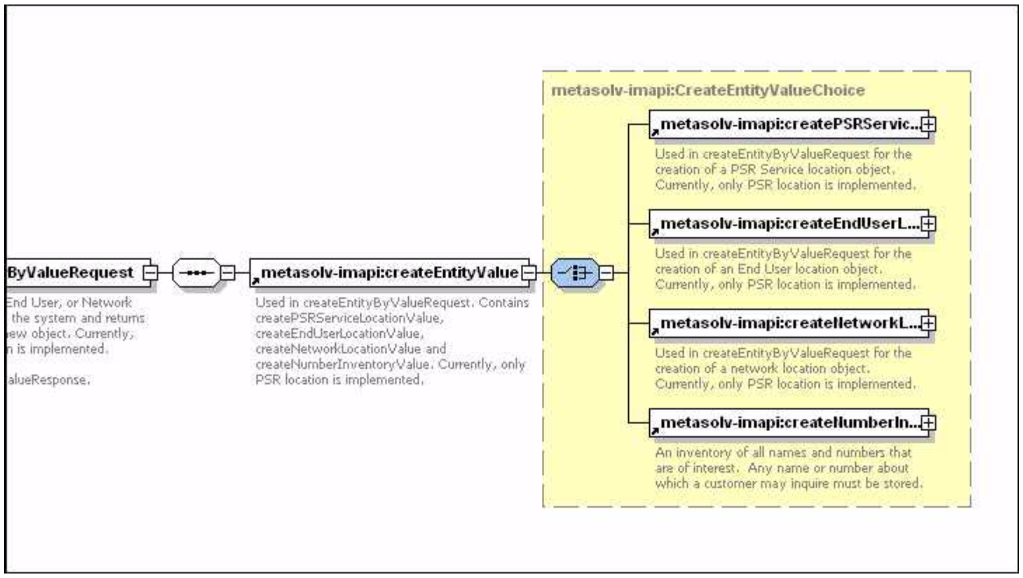
4. You are now viewing the contents of the createEntityByValueReqeust structure. Further expand the createEntityByValueReqeust structure by clicking on the expand button as indicated below.


createEntityByValueRequest
metasolv-imapi:createEntityValue
metasolv-imapi:CreateEntityValueChoice

Creates a PSR, End User, or Network location object in the system and returns the key for the new object. Currently, only PSR location is implemented. Response: createEntityByValueResponse.

Used in createEntityByValueRequest. Contains createPSRServiceLocationValue, createEndUserLocationValue, createNetworkLocationValue and createNumberInventoryValue. Currently, only PSR location is implemented.

5. You are now viewing sub-structures that are a choice. For example, if you wish to create a PSR Service Location you would populate the first choice of sub-structures with input data; if you wish to create a Network Location you would populate the third choice of sub-structures with input data. Do not include the remaining empty choice structures in the request.


ByValueRequest
metasolv-imapi:createEntityValue
metasolv-imapi:CreateEntityValueChoice
metasolv-imapi:createPSRServic...
metasolv-imapi:createEndUserL...
metasolv-imapi:createNetworkL...
metasolv-imapi:createNumberIn...

End User, or Network
the system and returns
ew object. Currently,
n is implemented.

alueResponse.

Used in createEntityByValueRequest. Contains createPSRServiceLocationValue, createEndUserLocationValue, createNetworkLocationValue and createNumberInventoryValue. Currently, only PSR location is implemented.

Used in createEntityByValueRequest for the creation of a PSR Service location object. Currently, only PSR location is implemented.

Used in createEntityByValueRequest for the creation of an End User location object. Currently, only PSR location is implemented.

Used in createEntityByValueRequest for the creation of a network location object. Currently, only PSR location is implemented.

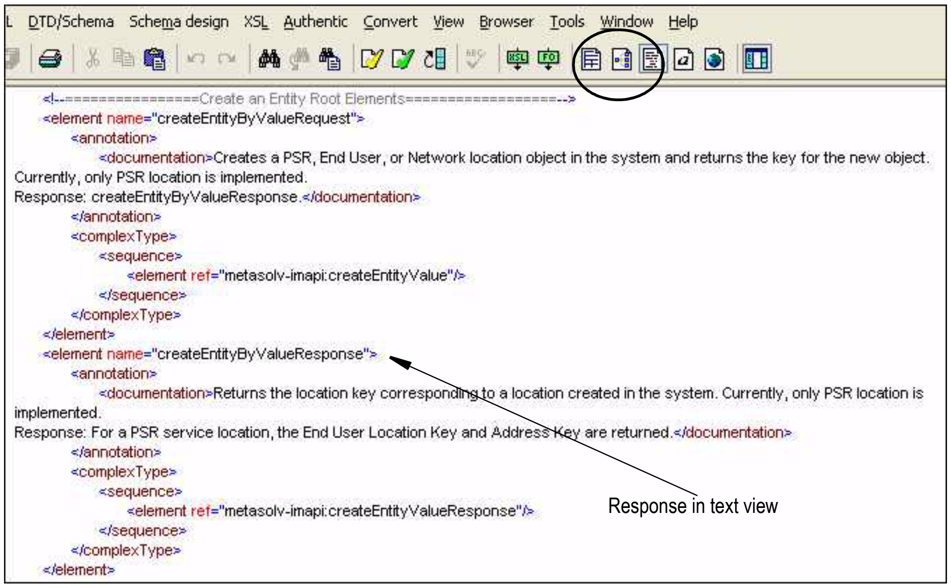
An inventory of all names and numbers that are of interest. Any name or number about which a customer may inquire must be stored.

Response structure

The method defines it's return as type com.metasolv.mip.inventoryManagementAPI.CreateEntityByValueResponseDocument. This tells us that an xml structure named *createEntityByValueResponse* is defined in the InventoryManagementAPI.xsd. Therefore, we will examine the xml structure *createEntityByValueResponse*, which is what is returned by the control method createEntityByValueRequest.

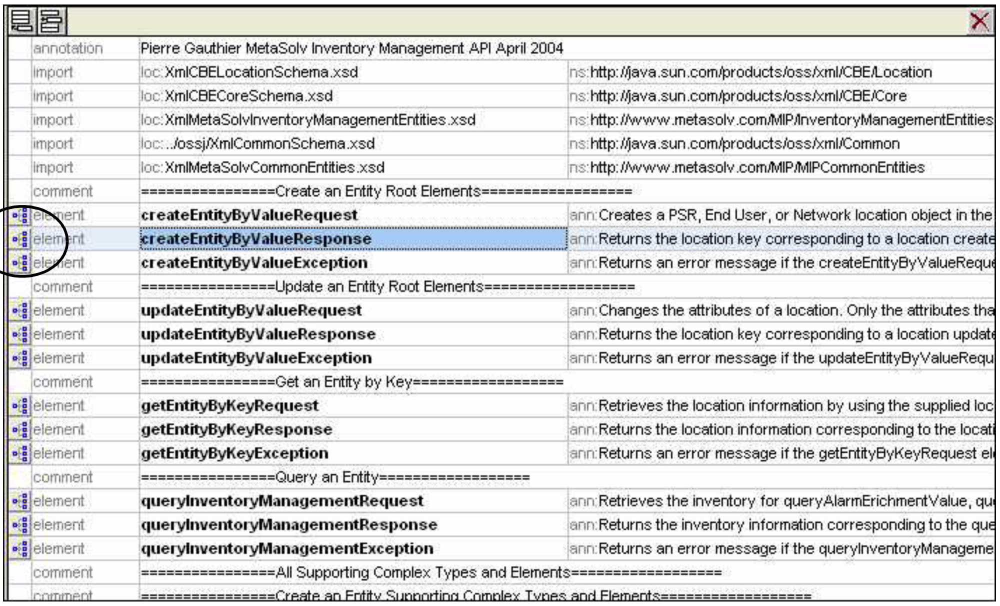
The following steps will walk you through viewing and understanding the CreateEntityByValueRequest structure.

1. Open the InventoryManagementAPI.xsd, using an xml tool such as XMLSpy.
2. If the xml file comes up as text view, change the view by selecting a different view as shown below.


L DTD/Schema Schema design XSL Authentic Convert View Browser Tools Window Help
Response in text view

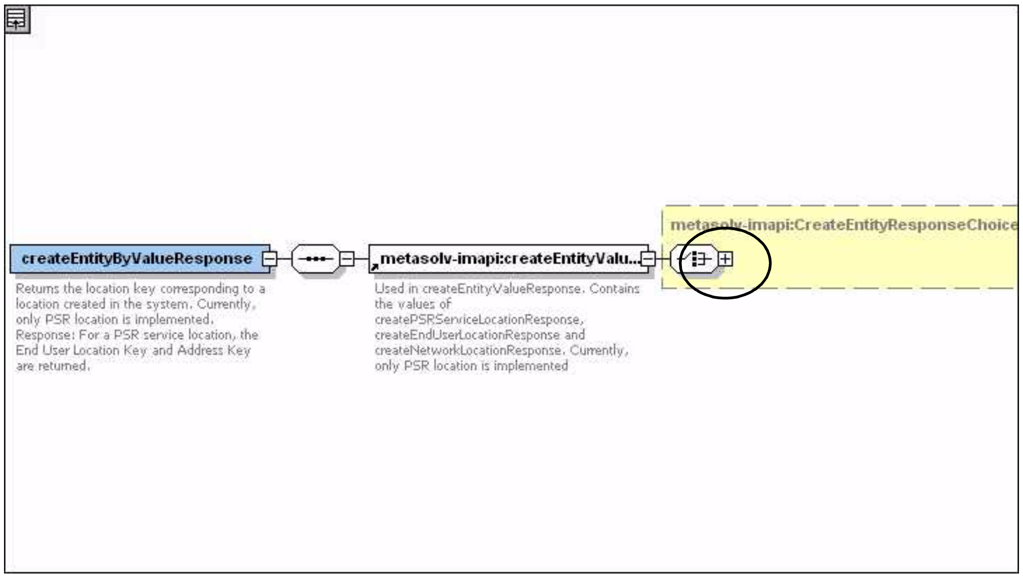
```
    <!--================Create an Entity Root Elements===================-->
    <element name="createEntityByValueRequest">
        <annotation>
            <documentation>Creates a PSR, End User, or Network location object in the system and returns the key for the new object.
Currently, only PSR location is implemented.
Response: createEntityByValueResponse.</documentation>
        </annotation>
        <complexType>
            <sequence>
                <element ref="metasolv-imapi:createEntityValue"/>
            </sequence>
        </complexType>
    </element>
    <element name="createEntityByValueResponse">
        <annotation>
            <documentation>Returns the location key corresponding to a location created in the system. Currently, only PSR location is
implemented.
Response: For a PSR service location, the End User Location Key and Address Key are returned.</documentation>
        </annotation>
        <complexType>
            <sequence>
                <element ref="metasolv-imapi:createEntityValueResponse"/>
            </sequence>
        </complexType>
    </element>
```

3. The top-level structures are now clearly listed. Expand the createEntityByValueResponse structure by clicking on the expand button as indicated below.



annotation	Pierre Gauthier MetaSolv Inventory Management API April 2004	
import	loc: XmlCBELocationSchema.xsd	ns: http://java.sun.com/products/oss/xml/CBE/Location
import	loc: XmlCBECoreSchema.xsd	ns: http://java.sun.com/products/oss/xml/CBE/Core
import	loc: XmlMetaSolvInventoryManagementEntities.xsd	ns: http://www.metasolv.com/MIP/InventoryManagementEntities
import	loc: ../ossj/XmlCommonSchema.xsd	ns: http://java.sun.com/products/oss/xml/Common
import	loc: XmlMetaSolvCommonEntities.xsd	ns: http://www.metasolv.com/MIP/MIPCommonEntities
comment	================Create an Entity Root Elements==================	
element	**createEntityByValueRequest**	ann: Creates a PSR, End User, or Network location object in the
element	**createEntityByValueResponse**	ann: Returns the location key corresponding to a location create
element	**createEntityByValueException**	ann: Returns an error message if the createEntityByValueReque
comment	================Update an Entity Root Elements==================	
element	**updateEntityByValueRequest**	ann: Changes the attributes of a location. Only the attributes tha
element	**updateEntityByValueResponse**	ann: Returns the location key corresponding to a location update
element	**updateEntityByValueException**	ann: Returns an error message if the updateEntityByValueRequ
comment	================Get an Entity by Key==================	
element	**getEntityByKeyRequest**	ann: Retrieves the location information by using the supplied loc
element	**getEntityByKeyResponse**	ann: Returns the location information corresponding to the locati
element	**getEntityByKeyException**	ann: Returns an error message if the getEntityByKeyRequest el
comment	================Query an Entity==================	
element	**queryInventoryManagementRequest**	ann: Retrieves the inventory for queryAlarmErichmentValue, qu
element	**queryInventoryManagementResponse**	ann: Returns the inventory information corresponding to the que
element	**queryInventoryManagementException**	ann: Returns an error message if the queryInventoryManageme
comment	================All Supporting Complex Types and Elements==================	
comment	================Create an Entity Supporting Complex Types and Elements==================	

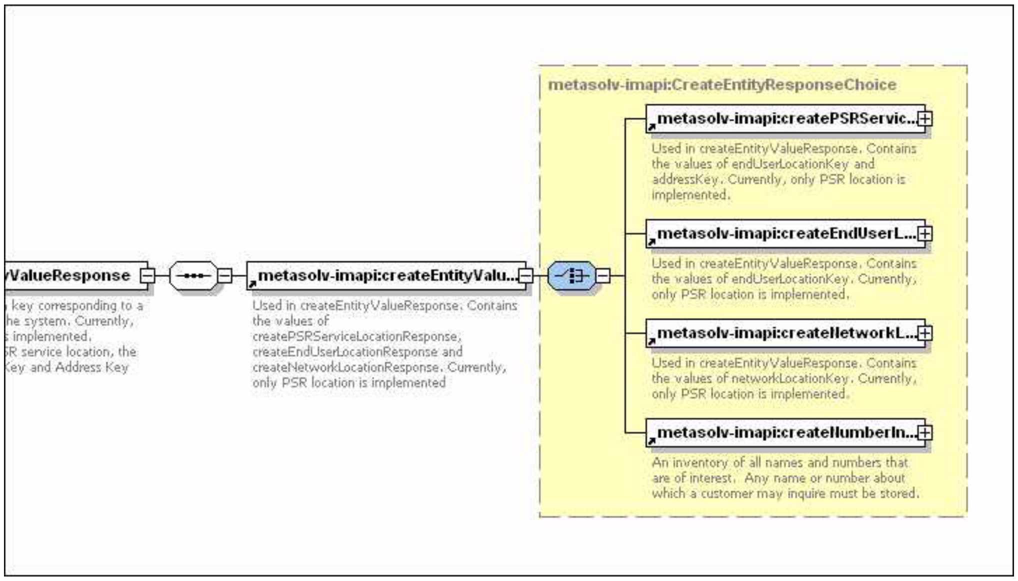
4. You are now viewing the contents of the createEntityByValueResponse structure. Further expand the createEntityByValueResponse structure by clicking on the expand button as indicated below.


createEntityByValueResponse
metasolv-imapi:createEntityValu...
metasolv-imapi:CreateEntityResponseChoice

Returns the location key corresponding to a location created in the system. Currently, only PSR location is implemented. Response: For a PSR service location, the End User Location Key and Address Key are returned.

Used in createEntityValueResponse. Contains the values of createPSRServiceLocationResponse, createEndUserLocationResponse and createNetworkLocationResponse. Currently, only PSR location is implemented

5. You are now viewing sub-structures that are a choice. For example, if you created a PSR Service Location, the first choice of sub-structures will be populated in the response; if you created a Network Location, the third choice of sub-structures will be populated in the in the response. Only one of the sub-structures will be returned in response.


ValueResponse
metasolv-imapi:createEntityValu...
metasolv-imapi:CreateEntityResponseChoice
metasolv-imapi:createPSRServic...
metasolv-imapi:createEndUserL...
metasolv-imapi:createNetworkL...
metasolv-imapi:createNumberIn...

key corresponding to a
he system. Currently,
s implemented.
SR service location, the
Key and Address Key

Used in createEntityValueResponse. Contains the values of createPSRServiceLocationResponse, createEndUserLocationResponse and createNetworkLocationResponse. Currently, only PSR location is implemented

Used in createEntityValueResponse. Contains the values of endUserLocationKey and addressKey. Currently, only PSR location is implemented.

Used in createEntityValueResponse. Contains the values of endUserLocationKey. Currently, only PSR location is implemented.

Used in createEntityValueResponse. Contains the values of networkLocationKey. Currently, only PSR location is implemented.

An inventory of all names and numbers that are of interest. Any name or number about which a customer may inquire must be stored.

Appendix D: XML API methods

The methods in this appendix are presented in the order they appear in the control file. The information provided for each method will include:

- Control name
- xml defined Request name
- xml defined Response name
- Description of method
- Input Structure
- Response Structure
- MetaSolv Solution Path > Page (where applicable)
- Additional information (where applicable)

Customer Management API

The Customer Management API is a collection of methods that provide the ability to import and maintain customer accounts in the MetaSolv Solution database from an outside source, without using the MetaSolv Solution GUI. The XmlMetaSolvCustomerManagementAPI.xsd defines the following methods:

importCustomerAccount

updateCustomerAccountByValueRequest

updateCustomerAccountByValueResponse

Description	This method either creates or updates the specified customer account based on the input data	
Input Structure	MetaSolvCustomerAccountValue Choice	MetaSolvCustomerAcountValue
Response Structure	MetaSolvCustomerAccountKey	
MetaSolv Solution Path > Page	Order Management > Customer Account	

Table 2: importCustomerAccount

Additional Information

This method provides the ability to import a new customer account, and to update an existing customer account. When the customerNr and suppType are not populated, the code processes the request as an import. When the customerNr and suppType are populated, the code processes the request as an update. When importing a new customer, the customerNr and suppType must not be populated. Also, importing a new customer requires additional data that is not required for an update.

getCustomerAccountByKey

getCustomerAccountByKeyRequest

getCustomerAccountByKeyResponse

Description	This method returns existing customer account information based on the input customer account key.	
Input Structure	MetaSolvCustomerAccountKey Choice	MetaSolvCustomerAccountKey
Response Structure	MetaSolvCustomerAccountValue Choice	MetaSolvCustomerAccountValue
MetaSolv Solution Path > Page	Order Management > Customer Account	

Table 3: getCustomerAccountByKey

deleteCustomerRequest

deleteCustomerAccountByKeyRequest

deleteCustomerAccountByKeyResposne

Description	This method deletes an existing customer account based on the input customer account key.	
Input Structure	MetaSolvCustomerAccountKey Choice	MetaSolvCustomerAccountKey
Response Structure	status (String)	
MetaSolv Solution Path > Page	Order Management > Customer Account	

Table 4: deleteCustomerRequest

Order Management API

The Order Management API is a collection of methods that provide the ability to import and maintain orders in the MetaSolv Solution database from an outside source, without using the MetaSolv Solution GUI. Orders include Internal Service Requests (ISRs) and Product Service Requests (PSRs). Local Service Requests (LSRs) are handled by a seperate API, Local Service Request API. The XmlMetaSolvOrderManagementAPI.xsd defines the following methods:

queryOrderManagementRequest

queryOrderManagementRequest

queryOrderManagementResponse

Description	This method returns various order information based on the choice of input structure.	
Input Structure	MetaSolvQueryValueChoice	ValidateOrderQueryValue, GetTaskGWEventQueryValue, GetServReqTasksQueryValue, GetServiceRequestDLRsValue, GetDLRInfosByOrderAndService ItemIdValue, GetDLRInfosByServiceItemIdIn ServiceValue, GetServItemReferenceValue, GetServItemsValue, GetProductCatalog,or GetOrderStatus
Response Structure	MetaSolvQueryResponseChoice	ValidateOrderQueryResponse, GetTaskGWEventQueryResponse, GetServReqTasksQueryResponse, GetServiceRequestDLRsResponse, GetDLRInfosByOrderAndServiceIt emIdValue, GetDLRInfosByServiceItemIdIn ServiceValue, GetServItemReferenceValue, GetServItemsValue, GetDLRInfosByOrderAndServiceIt emIdResponse, GetDLRInfosByServiceItemIdInSer viceResponse, GetServItemReferenceResponse, GetServItemsResponse, GetProductCatalogResponse, or GetOrderStatusResponse
MetaSolv Solution Path > Page		

Table 5: queryOrderManagementRequest

startOrderByKeyRequest

startOrderByKeyRequest

startOrderByKeyResponse

Description	This method initiates the ‘Finish Order’ processing for the input order.	
Input Structure	MetaSolvOrderKeyChoice	OrderKey
Response Structure	MetaSolvOrderKeyChoice	OrderKey
MetaSolv Solution Path > Page	Order Management > Product Service Request or Order Management > Internal Service Request	

Table 6: startOrderByKeyRequest

Additional Information

This method needs to be called after a successful response from createOrderByValueRequest, and before calling assignProvisionPlanProcedureRequest. It is the API equivalent of clicking the GUI link for "Finish Order".

updateOrderManagementRequest

updateOrderManagementRequest

updateOrderManagementResponse

Description	This method updates various order information, based on the choice input structure and the input data.	
Input Structure	MetaSolvUpdateProcedureValueChoice	UpdateServicesInOrderProcedureValue, UpdateOrderTaskGWEventValue, CompleteTaskProcedureValue, UpdateOrderTaskEventProcedureValue, or ReopenTaskProcedureValue
Response Structure	MetaSolvUpdateProcedureValueResponseChoice	UpdateServicesInOrderProcedureResponse, UpdateOrderTaskGWEventResponse, CompleteTaskProcedureResponse, UpdateOrderTaskEventProcedureResponse, or ReopenTaskProcedureResponse
MetaSolv Solution Path > Page		

Table 7: updateOrderManagementRequest

getOrderByKeyRequest

getOrderByKeyRequest

getOrderByKeyResponse

Description	This method returns order information based on the input order key.	
Input Structure	MetaSolvOrderKeyChoice	OrderKey
Response Structure	MetaSolvOrderValueChoice	MetaSolvPSROrderValue or MetaSolvISROrderValue
MetaSolv Solution Path > Page	Order Management > Product Service Request or Order Management > Internal Service Request	

Table 8: getOrderByKeyRequest

createOrderByValueRequest

createOrderByValueRequest

createOrderByValueResponse

Description	This method creates a new PSR or ISR order based on the input data.	
Input Structure	MetaSolvOrderValueChoice	MetaSolvPSROrderValue or MetaSolvISROrderValue
Response Structure	MetaSolvOrderKeyChoice	OrderKey
MetaSolv Solution Path > Page	Order Management > Product Service Request or Order Management > Internal Service Request	

Table 9: createOrderByValueRequest

Additional Information

This method will create a PSR order or an ISR order, depending on the choice of the input structure. Both input structures define the same sub-structure, OrderHeaderType, which is where the mutual required data is defined.

assignProvisionPlanProcedureRequest

assignProvisionPlanProcedureRequest

assignProvisionPlanProcedureResponse

Description	This method assigns a provisioning plan to an order based on the input data.	
Input Structure	AssignProvisionPlanProcedure Value	ProvisionPlanValue
Response Structure	MetaSolvOrderKeyChoice	OrderKey
MetaSolv Solution Path > Page		

Table 10: assignProvisionPlanProcedureRequest

getActivationDataByKeyRequest

getActivationDataByKeyRequest

getActivationDataByKeyResponse

Description	This method returns activation information based on the input order key and service key.	
Input Structures	OrderKey MetaSolvServiceKey	
Response Structure	MetaSolvServiceActivationType	
MetaSolv Solution Path > Page		

Table 11: getActivationDataByKeyRequest

transferTaskRequest

transferTaskRequest

transferTaskResponse

Description	This method transfers tasks between work queues based on the input data.	
Input Structure	TransferTaskValueType	orderKey, taskNumber, currentWorkQueue, newWorkQueue
Response Structure	OrderKey	
MetaSolv Solution Path > Page		

Table 12: tranferTaskRequest

updateE911DataRequest

updateE911DataReqest

updateE911DataResponse

Description	This method updates E911 data based upon the input data.	
Input Structure	E911DataType	
Response Structure		
MetaSolv Solution Path > Page		

Table 13: updateE911DataRequest

getE911DataRequest

getE911DataRequest

getE911DataResponse

Description	This method returns E911 data, based upon the input data.	
Input Structure		
Response Structure	E911DataType	
MetaSolv Solution Path > Page		

Table 14: getE911DataRequest

updateEstimationCompletedDateRequest

updateEstimatedCompletionDateRequest

updateEstimatedCompletionDateResponse

Description	This method updates the estimated completion dates of tasks based on the input order and specified tasks associated with the order.	
Input Structure	UpdateEstimatedCompletionDate ValueType	
Response Structure	status (String)	
MetaSolv Solution Path > Page		

Table 15: updateEstimationCompletedDateRequest

addTaskJeopardyRequest

addTaskJeopardyRequest

addTaskJeopardyResponse

Description	This method adds task jeopardy information based on the input task data.	
Input Structure	AddTaskJeopardyRequestValue Type	
Response Structure	status (String)	
MetaSolv Solution Path > Page		

Table 16: addTaskJeopardyRequest

getTaskDetailRequest

getTaskDetailRequest

getTaskDetailResponse

Description	This method returns task detail information based on the input data.	
Input Structure	GetTaskDetailRequestValueType	
Response Structure	GetTaskDetailResponseValue Type	
MetaSolv Solution Path > Page		

Table 17: getTaskDetailRequest

TaskJeopardyRequest

getTaskJeopardyRequest

getTaskJeopardyResponse

Description	This method returns task jeopardy information based on the input task data.	
Input Structure	GetTaskJeopardyRequestValue Type	
Response Structure	GetTaskJeopardyResponseValueT ype	
MetaSolv Solution Path > Page		

Table 18: TaskJeopardyRequest

getPSROrderByTN

getPSROrderByTNRequest

getPSROrderByTNResponse

Description	This method returns PSR order information based on the input telephone number.	
Input Structure	GetPSROrderByTNRequestValue Type	
Response Structure	MetaSolvOrderValueChoice	MetaSolvPSROrderValue or MetaSolvISROrderValue
MetaSolv Solution Path > Page		

Table 19: getPSROrderByTN

processSuppOrder

processSuppOrderRequest

processSuppOrderResponse

Description	This method processes a supplement order based on the input data.	
Input Structure	ProcessSuppOrderRequestValue Type	
Response Structure	message (String)	
MetaSolv Solution Path > Page		

Table 20: processSuppOrder

getCNAMDataRequest

getCNAMDataRequest

getCNAMDataResponse

Description	This method returns CNAM data based upon the input data.	
Input Structure		
Response Structure	CNAMDataType	
MetaSolv Solution Path > Page		

Table 21: getCNAMDataRequest

getLidbDataRequest

getLIDBDataRequest

getLIDBDataResponse

Description	This method returns LIDB data based on the input data.	
Input Structure		
Response Structure	LIDBDataType	
MetaSolv Solution Path > Page		

Table 22: getLidbDataRequest

updateCNAMDataRequest

updateCNAMDataRequest

updateCNAMEDataResponse

Description	This method updates CNAM data based on the input data.	
Input Structure	CNAMDataType	
Response Structure	status (String)	
MetaSolv Solution Path > Page		

Table 23: updateCNAMDataRequest

updateLidbDataRequest

updateLIDBDataRequest

updateLIDBDataResponse

Description	This method updates LIDB data based on the input data.	
Input Structure	LIDBDataType	
Response Structure	status (String)	
MetaSolv Solution Path > Page		

Table 24: updateLidbDataRequest

reopenTaskRequest

reopenTaskRequest

reopenTaskResponse

Description	This method reopens a task based on the input data.	
Input Structure	ReopenTaskValueType	
Response Structure	MetaSolvOrderValueChoice	MetaSolvPSROrderValue or MetaSolvISROrderValue
MetaSolv Solution Path > Page		

Table 25: reopenTaskRequest

createAttachmentRequest

createAttachmentRequest

createAttachmentResponse

Description	This method creates an xml document attachment that is associated with a PSR based on the input data.	
Input Structure	CreateAttacmentType	
Response Structure	message (String)	
MetaSolv Solution Path > Page		

Table 26: createAttachmentRequest

createOrderRelationshipRequest

createOrderRelationshipRequest

createOrderRelationshipResponse

Description	This method creates a parent/child relationship based on the two input orders.	
Input Structures	OrderKey (parent) OrderKey (child)	
Response Structure	message (String)	
MetaSolv Solution Path > Page	Service Request>Service Request Hierarchy	

Table 27: createOrderRelationshipRequest

Additional Information

This method only supports the order relationship type of parent/child. This is important to note because MetaSolv Solution supports several different relationship types, but this method only supports the relationship type of parent/child.

processBillingTelephoneNumber

billingTelephoneNumber

billingTelephoneNumberResponse

Description	This method receives a structure to be passed to process the number as Billing Telephone Number.	

Input Structures	documentNumber (long)
	servItemId(long)
	BtnFunction-Enum(enum defined in same xsd file, as a String with value of 0 or 1)
	nbrInvId(long)	

Response Structure	documentNumber (integer)	
MetaSolv Solution Path > Page	Order Management > Product Service Request	

Table 28: processBillingTelephoneNumber

Inventory Management API

The Inventory Management API is a collection of methods that provide the ability to import and maintain inventory in the MetaSolv Solution database from an outside source, without using the MetaSolv Solution GUI. Inventory items include Locations (PSR Service Locations, End User Locations, and Network Locations), Telephone Numbers, Physical Ports, and IP Addresses. The XmlMetaSolvInventoryManagementAPI.xsd defines the following methods:

createEntityByValueRequest

createEntityByValueRequest

createEntityByValueResponse

Description	This method creates a new entity based on the choice of input structure.	
Input Structure	CreateEntityValueChoice	CreatePSRServiceLocationValue, CreateEndUserLocationValue, CreateNetworkLocationValue, or CreateNumberInventoryValue
Response Structure	CreateEntityResponseChoice	CreatePSRServiceLocation Response, CreateEndUserLocationResponse, CreateNetworkLocationResponse, or CreateNumberInventoryResponse
MetaSolv Solution Path > Page		

Table 29: createEntityByValueRequest

getServiceRequestDLRsValue

createEntityByValueRequest

createEntityByValueResponse

Description		
Input Structure		
Response Structure		
MetaSolv Solution Path > Page		

Table 30: getServiceRequestDLRsValue

getEntityByKeyRequest

getEntityByValueRequest

getEntityByValueResponse

Description	This method returns various entity information based on the choice of input structure.	
Input Structure	GetEntityByKeyValueChoice	GetPSRServiceLocationByKey, GetEndUserLocationByKey, GetNetworkLocationByKey, GetDlrByKey
Response Structure	GetEntityByKeyResponseChoice	GetPSRServiceLocationResponse GetEndUserLocationResponse, GetNetworkLocationResponse, GetDlrByKeyResponse
MetaSolv Solution Path > Page		

Table 31: getEntityByKeyRequest

updateEntityByValueRequest

updateEntityByValueRequest

updateEntityByValueResponse

Description	This method updates an existing entity based on the choice of input structure.	
Input Structure	UpdateEntityValueChoice	UpdatePSRServiceLocationValue or UpdatePreAssignTelephone NumberValue
Response Structure	UpdateEntityResponseChoice	UpdatePSRServiceLocation Response or UpdatePreAssignTelephone NumberResponse
MetaSolv Solution Path > Page		

Table 32: updateEntityByValueRequest

queryInventoryManagementRequest

queryInventoryManagementRequest

queryInventoryManagementResponse

Description	This method returns various inventory management information based on the choice of input structure.	
Input Structure	MetaSolvInventoryQueryValue Choice	QueryAlarmEnrichmentValue, QueryEquipmentCapacityValue, QueryTelephoneNumber InventoryValue, QueryMSAGInventoryValue
Response Structure	MetaSolvInventoryQuery ResponseChoice	QueryAlarmEnrichmentResponse, QueryEquipmentCapacity Response, QueryTelephoneNumber InventoryResponse, QueryMSAGResponse
MetaSolv Solution Path > Page		

Table 33: queryInventoryManagementRequest

updateTNRequest

updateTNRequest

updateTNResponse

Description	This method updates an existing telephone number based on the input data.	
Input Structure	UpdateTNRequestValueType	
Response Structure	UpdateTNResponseValue (int)	
MetaSolv Solution Path > Page		

Table 34: updateTNRequest

tnRecall

processTNRecallRequest

processTNRecallResponse

Description	This method recalls a telephone number based on the input data.	
Input Structure	tn (String)	
Response Structure	ProcessTNRecallResponseValue Type	
MetaSolv Solution Path > Page		

Table 35: tnRecall

tnValidationRequest

processTNValidationRequest

processTNValidationResponse

Description	This method validates the TN.	
Input Structure	tn (String)	
Response Structure	ProcessTNValidationResponse Value (String)	
MetaSolv Solution Path > Page		

Table 36: tnValidationRequest

auditTrailRecording

Request

Response

Description		
Input Structure		
Response Structure		
MetaSolv Solution Path > Page		

Table 37: auditTrailRecording

getNetworkAreasByGeoAreaRequest

getNetworkAreasByGeoAreaRequest

getNetworkAreasByGeoAreaResponse

Description	This method returns Network Area information based on the input Geographical Area.	
Input Structure	GeoAreaCriteria	
Response Structure	NetworkArea	
MetaSolv Solution Path > Page		

Table 38: getNetworkAreasByGeoAreaRequest

getNetworkComponentsRequest

getNetworkComponentsRequest

getNetworkComponentsResponse

Description	This method returns Network Component information based on the input data.	
Input Structure	getNetworkComponentsRequest ValueType	
Response Structure	NetworkComponent	
MetaSolv Solution Path > Page		

Table 39: getNetworkComponentsRequest

getIpAddressesRequest

getIpAddressesRequest

getIpAddressesResponse

Description	This method returns ip address information based on the input data.	
Input Structure	IpAddressCriteria	
Response Structure	IpAddressesValue (sequence)	
MetaSolv Solution Path > Page		

Table 40: getIpAddressesRequest

createInventoryAssociationRequest

createInventoryAssociationRequest

createInventoryAssociationResponse

Description	This creates a relationship between two inventory items based on the input data.	
Input Structure	ImportInventoryAssociation	
Response Structure	status	
MetaSolv Solution Path > Page		

Table 41: createInventoryAssociationRequest

createNewInventoryItemRequest

createNewInventoryItemRequest

createNewInventoryItemResponse

Description	This method creates a new inventory item based on the input data.	
Input Structure	InventoryItem	
Response Structure	MetaSolvNumberInventoryKey	
MetaSolv Solution Path > Page		

Table 42: createNewInventoryItemRequest

queryNetworkLocation

queryNetworkLocationRequest

queryNetworkLocationResponse

Description	This method retrieves multiple network locations from the MSS database based on the input data.	
Input Structure	NetworkLocationQueryValue	
Response Structure	NetworkLocationResultValue	
MetaSolv Solution Path > Page		

Table 43: queryNetworkLocation

queryEndUserLocation

queryEndUserLocationRequest

queryEndUserLocationResponse

Description	This method retrieves multiple end user locations from the MSS database based on the input data.	
Input Structure	EndUserLocationQueryValue	
Response Structure	EndUserLocationResultValue	
MetaSolv Solution Path > Page		

Table 44: queryEndUserLocation

getLocationRequest

getLocationRequest

getLocationResponse

Description	This method retrieves a specific location from the MSS database based on the input data key.	
Input Structure	NetworkLocationKey	
Response Structure	LocationValue	
MetaSolv Solution Path > Page		

Table 45: getLocationRequest

deleteLocationRequest

deleteLocationByKey

deleteLocationResponse

Description	This method deletes a specific location from the MSS database based on the input data key.	
Input Structure	NetworkLocationKey	
Response Structure	NetworkLocationKey	
MetaSolv Solution Path > Page		

Table 46: deleteLocationRequest

updateLocationRequest

updateLocationRequest

updateLocationResponse

Description	This method updates a specific location in the MSS database based on the input data.	
Input Structure	LocationValue	
Response Structure	NetworkLocationKey	
MetaSolv Solution Path > Page		

Table 47: updateLocationRequest

createLocationRequest

createLocationRequest

createLocationResponse

Description	This method creates a new location in the MSS data base based on the input data.	
Input Structure	LocationValue	
Response Structure	NetworkLocationKey	
MetaSolv Solution Path > Page		

Table 48: createLocationRequest

getAvailablePhysicalPortsRequest

getAvailablePhysiclaPortsRequest

getAvailablePhysicalPortsResponse

Description	This method returns a sequence of available physical ports based on the input data.	
Input Structure	getAvailablePhysicalPortsRequest ValueType	
Response Structure	PhysicalPort (sequence)	
MetaSolv Solution Path > Page		

Table 49: getAvailablePhysicalPortsRequest

Additional Information

Note that this method is defined in a different control class than the rest of the methods defined in this section. While the InventoryManagementAPI.xsd defines the structures for this method, the InventoryManagement control class does not define the control. Rather, the control is defined in the NetworkResourceManagement control class. The NetworkResourceManagement will continue to grow in future releases, at which point a new xsd will be created.

Service Order Activation API

The Service Order Activation API is a collection of methods that provide the ability to activate services, previously placed on orders, in the MetaSolv Solution database from an outside source, without using the MetaSolv Solution GUI. The XmlMetaSolvInventoryManagementAPI.xsd defines the following methods:

createSOAMessageRequest

createSOAMessageRequest

createSOAMessageResponse

Description	This method creates a SOA message based on the input data.	
Input Structure	SOATransactionType, OrderKey	
Response Structure	SOATransactionKey	
MetaSolv Solution Path > Page		

Table 50: createSOAMessageRequest

getSoaTnsForOrderRequest

getSOATNsForOrderRequest

getSOATNsForOrderResponse

Description	This method returns SOA telephone numbers based on the input order.	
Input Structure	OrderKey	
Response Structure	SOATelephoneNumberType (sequence)	
MetaSolv Solution Path > Page		

Table 51: getSoaTnsForOrderRequest

getSoaDefaultsRequest

getSOADefaultsRequest

getSOADefaultsResponse

Description	This method returns the SOA defaults based on the input data.	
Input Structure	OrderKey, SOATelephoneNumberType	
Response Structure	SOADefaultsType	
MetaSolv Solution Path > Page		

Table 52: getSoaDefaultsRequest

getSoaInformationRequest

getSOAInformationRequest

getSOAInformationResponse

Description	This method returns SOA information based on the input data.	
Input Structure	OrderKey, SOATelephoneNumberType	
Response Structure	SOAInformationType	
MetaSolv Solution Path > Page		

Table 53: getSoaInformationRequest

getSoaMessageToSendRequest

getSOAMessagesToSendRequest

getSOAMessagesToSendResponse

Description	This method returns SOA messages to send based on the input data.	
Input Structure	OrderKey, checkGatewayEventReactivated (boolean)	
Response Structure	SOATransactionType (sequence)	
MetaSolv Solution Path > Page		

Table 54: getSoaMessagesToSendRequest

setTnSoaCompleteRequest

setTNSOACompleteRequest

setTNSOACompleteResponse

Description	This method sets SOA for the telephone number to complete.	
Input Structure	OrderKey, SOATelephoneNumberType	
Response Structure	successfulCompletion (boolean)	
MetaSolv Solution Path > Page		

Table 55: setTnSoaCompleteRequest