SYBASE®

SybStore Tutorials: Service Development and Process Orchestration

Sybase® WorkSpace

1.5

DOCUMENT ID: DC00494-01-0150-01

LAST REVISED: June 2006

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

About This Book ..... **vii**

CHAPTER 1 **Introduction, Installation, and Setup** **1**
- Introduction 1
  - SybStore tutorials 2
  - SybStore sample 3
- Installing Sybase WorkSpace 3
  - Starting and exploring Sybase WorkSpace 4
  - Becoming familiar with the Eclipse environment 5
- Setting up the SybStore tutorial 6
  - Downloading the MySybStore_Tutorials project 6
  - Importing the tutorial files into Sybase WorkSpace 7
  - Starting and connecting to the MySybStore database 12
  - Initializing the tutorial database 18
  - Starting and connecting to the Unwired Orchestrator server .. 19

CHAPTER 2 **Service Development Tutorials** **21**
- Overview 21
  - Prerequisites 22
- Creating a database service 22
  - Lesson 1: Creating a database service 22
- Creating a Java service 27
  - Lesson 1: Creating a Java service 27
  - Lesson 2: Invoking a database service from Java service operation 37
- Creating a transformation service 43
  - Lesson 1: Creating a new transformation service 43
  - Lesson 2: Defining mapping for a transformation service 51
  - Lesson 3: Testing a transformation service 62
- Creating a message service 64
  - Lesson 1: Creating a message service 65
  - Lesson 2: Verifying service parameters 67
  - Lesson 3: Specifying e-mail message fields 70

Using a generic JMS provider for messaging 72
Lesson 1: Specifying the location of generic JMS client JAR files 73
Lesson 2: Configuring generic JMS transport connection profile . 75
Lesson 3: Creating a generic JMS transport messaging service . 78
Packaging, deploying, and testing a service 79
Lesson 1: Creating a package profile 80
Lesson 1: Changing the logging level and packaging the service 81
Lesson 3: Deploying a service 83
Lesson 4: Testing a service 85

CHAPTER 3 Process Orchestration Tutorials 89
Overview 89
Prerequisites 89
Creating a simple business process service 90
Lesson 1: Creating a business process service 90
Lesson 2: Adding a service invocation to a business process service 109
Lesson 3: Adding a rule to a business process service 120
Lesson 4: Defining error handling for a business process service 127
Lesson 5: Setting message context properties dynamically .. 134
Packaging, deploying, and testing a business process service ... 143
Lesson 1: Building the package 144
Lesson 2: Deploying the package 149
Lesson 3: Testing the service 149
Debugging a business process service 151
Lesson 1: Creating a Java service to write tracing to the log 151
Lesson 2: Adding tracing to a business process service 159
Creating a business process service correlation set 166
Lesson 1: Creating a business service to send a correlation request 167
Lesson 2: Creating a business process service correlation set ... 179
Lesson 3: Adding order-processing logic 180
Lesson 4: Adding logging activities 191
Lesson 5: Sending a correlated response to the initiating business process 195

CHAPTER 4 Unwired Orchestrator Logging Tutorials 203

Overview 203
Prerequisites 203
Using Unwired Orchestrator logging 204
Lesson 1: Setting the logging level and deploying a service. 204
Lesson 2: Executing a service to generate log data..... 209
Lesson 3: Importing the Unwired Orchestrator log file 212
Lesson 4: Reviewing the Unwired Orchestrator log file..... 215
Lesson 5: Viewing selected portions of the log file..... 218

CHAPTER 5 Cleaning up the Sybase WorkSpace environment 223
Closing active connections..... 223
Deleting the tutorial project 224
Recreating the tutorial project 224

About This Book

Audience

This document is for developers who want to use Sybase® WorkSpace integrated development tooling.

How to use this book

This guide is divided into these chapters:

- Chapter 1, “Introduction, Installation, and Setup,” introduces the Sybase WorkSpace product tutorials, and describes the tasks you must perform before you can run the tutorials.
- Chapter 2, “Service Development Tutorials,” illustrates how to use WorkSpace tools to create several types of services.
- Chapter 3, “Process Orchestration Tutorials,” shows you how to use WorkSpace tools to perform process orchestration using business process services.
- Chapter 4, “Unwired Orchestrator Logging Tutorials,” shows you how to use generate and view Unwired Orchestrator log data.
- Chapter 5, “Cleaning up the Sybase WorkSpace environment,” describes how to remove the tutorial files from your WorkSpace environment.

Related documents

Sybase WorkSpace tutorials and samples Sybase WorkSpace includes interactive tutorials and samples that show you how to use WorkSpace tools to create basic parts of a service-oriented application.

The tutorial and sample files and documentation are available for download from Sybase CodeXchange.

For more information about the tutorials and samples and instructions on how to download the files, select **Help**|**Tutorials** from the WorkSpace main menu bar. To get samples information, select the *Samples* Related Topic at the end of the *Tutorial* topic.

Sybase WorkSpace online bookshelf The WorkSpace online bookshelf contains all of the WorkSpace documentation. To access the WorkSpace bookshelf:

1 In Windows, select **Start**|**Programs**|**Sybase**|**Sybase WorkSpace**|**Sybase WorkSpace 1.5**.

2 Select **Help|Help Contents** from the WorkSpace main menu bar to open the main **Help** window.

The left pane displays the bookshelf contents, while the right pane displays the details of the selection in the left pane.

The WorkSpace bookshelf contains these document collections:

- *Sybase WorkSpace 1.5 What's New* – summarizes new functionality in this version.
- *Sybase WorkSpace Development* – includes Getting Started, and help for each major component service.
- *Sybase WorkSpace Server Administration* – documents how to stop, start, and manage the servers included with Sybase WorkSpace.

Sybase WorkSpace Getting Started CD The Sybase WorkSpace Getting Started CD includes these documents:

- *Sybase WorkSpace 1.5 Installation Guide*
- *Sybase WorkSpace 1.5 Release Bulletin*
- *Sybase Developer Edition Servers Installation Guide*
- *Sybase Adaptive Server Enterprise 15.0 Installation Guide*
- *Sybase Unwired Accelerator 7.0 Installation Guide*

Other sources of information

Use the Sybase Getting Started CD, the SyBooks™ CD, and the Sybase Product Manuals Web site to learn more about your product:

- The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the SyBooks CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader, which you can download at no charge from the Adobe Web site using a link provided on the CD.
- The SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access the manuals in an easy-to-use, HTML-based format.

  Some documentation may be provided in PDF format, which you can access through the PDF directory on the SyBooks CD. To read or print the PDF files, you need Adobe Acrobat Reader.

  Refer to the *SyBooks Installation Guide* on the Getting Started CD, or the *README.txt* file on the SyBooks CD for instructions on installing and starting SyBooks.

- The Sybase Product Manuals Web site is an online version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

  To access the Sybase Product Manuals Web site, go to Product Manuals at http://www.sybase.com/support/manuals/.

Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

❖ **Finding the latest information on product certifications**

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Select Products from the navigation bar on the left.
3. Select a product name from the product list and click Go.
4. Select the Certification Report filter, specify a time frame, and click Go.
5. Click a Certification Report title to display the report.

❖ **Finding the latest information on component certifications**

1. Point your Web browser to Availability and Certification Reports at http://certification.sybase.com/.
2. Either select the product family and product under Search by Product; or select the platform and product under Search by Platform.
3. Select Search to display the availability and certification report for the selection.

❖ **Creating a personalized view of the Sybase Web site (including support pages)**

Set up a MySybase profile. MySybase is a free service that allows you to create a personalized view of Sybase Web pages.

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Click MySybase and create a MySybase profile.

Sybase EBFs and software maintenance

❖ **Finding the latest information on EBFs and software maintenance**

1 Point your Web browser to the Sybase Support Page at http://www.sybase.com/support.

2 Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.

3 Select a product.

4 Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

Padlock icons indicate that you do not have download authorization for certain EBF/Maintenance releases because you are not registered as a Technical Support Contact. If you have not registered, but have valid information provided by your Sybase representative or through your support contract, click Edit Roles to add the "Technical Support Contact" role to your MySybase profile.

5 Click the **Info** icon to display the EBF/Maintenance report, or click the product description to download the software.

Conventions

The following formatting conventions are used in this document:

Formatting example	To indicate
command names and method names	When used in descriptive text, this font indicates keywords such as: • Command names used in descriptive text • C++ and Java method or class names used in descriptive text • Java package names used in descriptive text
myCounter variable `Server.log` *myfile.txt*	Italic font indicates: • Program variables • Parts of input text that must be substituted • Directory and file names
sybase\bin	A backward slash ("\") indicates cross-platform directory information. A forward slash ("/") applies to information specific only to UNIX.
File|Save	Menu names and menu items display in bold. The vertical bar indicates how to navigate menu selections, such as from the **File** menu to the **Save** option.

Formatting example	To indicate
`parse|put|get` `Name|Address`	In syntax and code examples, the vertical bar indicates: • Options available within code • Delimiter within message examples
`create table` `table created`	Monospace font indicates: • Information that you enter on a command line or as program text. • Example output fragments
Type the **Name** of the attribute. Click **Apply**.	GUI field or button name that is the recipient of a procedural action.
setup -is:tempdir <*full path to alternate temp directory*>	Information that must be supplied by the user is displayed between brackets.

Accessibility features

This document is available in an HTML version that is specialized for accessibility. You can navigate the HTML with an adaptive technology such as a screen reader, or view it with a screen enlarger.

Note You might need to configure your accessibility tool for optimal use. Some screen readers pronounce text based on its case; for example, they pronounce ALL UPPERCASE TEXT as initials, and MixedCase Text as words. You might find it helpful to configure your tool to announce syntax conventions. Consult the documentation for your tool.

For information about how Sybase supports accessibility, see Sybase Accessibility at http://www.sybase.com/accessibility. The Sybase Accessibility site includes links to information on Section 508 and W3C standards.

If you need help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

CHAPTER 1

Introduction, Installation, and Setup

This chapter introduces the Sybase WorkSpace Service Development and Process Orchestration tutorials, and describes the tasks you must perform before you begin the lessons.

Topic	Page
Introduction	1
Installing Sybase WorkSpace	3
Setting up the SybStore tutorial	6

Introduction

Sybase WorkSpace includes interactive tutorials that teach you how to use WorkSpace tooling to create basic parts of a service-oriented application. Each tutorial guides you through building a small component of the application.

The tutorials in this guide focus on WorkSpace Service Development and Process Orchestration tooling.

Service Development tooling allows you to encapsulate business logic and data into reusable networked software components. A service can be as simple as a credit check process involving a single request-reply operation, or as complicated as a business process that incorporate multiple services that access a variety of systems using multiple service types. Services are stored in Web Services Description Language (WSDL) files.

Process Orchestration allows you to orchestrate services into a business process to build an integration solution. After you create a business process service, you can add alerts, other services, business activity monitoring, custom wire formats, and database event management.

SybStore tutorials

The tutorials use the SybStore sample application, which is a sales and inventory system that automates the following retail business processes:

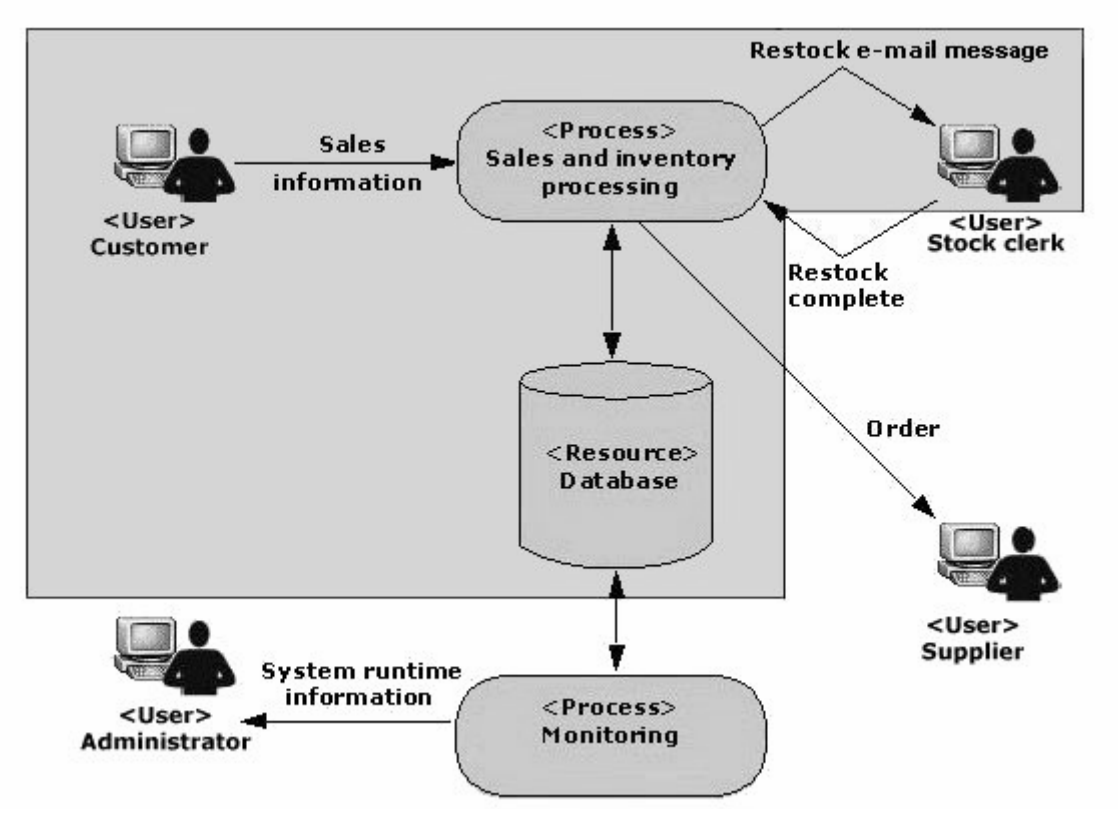
1 A customer buys items from the store, and the cash register records that the items have been removed from the shelves.

2 The sales and inventory system notifies the stock clerk on a PDA to restock the items.

3 The stock clerk receives an e-mail message (on his or her PDA) to restock specific items when the sales and inventory system determines that restocking is required.

4 When restocking is complete, the stock clerk updates the sales and inventory system, using the PDA.

The following illustration shows the basic flow of the SybStore application.



Note The illustration includes several actions that are not implemented in the SybStore tutorial application. The actions that are implemented in the SybStore tutorials are shown in the shaded block area and contain enough examples to demonstrate how to use Sybase WorkSpace.

Select **Help|Tutorials** from the WorkSpace main menu to learn about additional WorkSpace tutorials.

SybStore sample

Sybase WorkSpace includes a simple, services-based application samples that demonstrates modeling, database development, service development, mobile development, and Web application development.

You can refer to the SybStore sample applications at any time—before you start a tutorial, while you are working through a tutorial, or after you complete a tutorial—to explore the application component you manipulate in the tutorial, or to compare your results with the sample.

To download and use the component-based samples and documentation, see the online help topic *Sybase WorkSpace Development|Getting Started|Samples* for instructions.

Installing Sybase WorkSpace

To use the Service Development and Process Orchestration tutorials, install Sybase WorkSpace version 1.5 or Sybase WorkSpace 1.5 Evaluation software.

You must have these components installed before you begin the tutorials:

- Sybase WorkSpace Service Development and Process Orchestration tooling
- Adaptive Server® Anywhere 9.0.2 Developer Edition server
- Unwired Accelerator 7.0 Developer Edition (EAServer) server
- Unwired Orchestrator 5.1 Developer Edition server

See the *Sybase WorkSpace Installation Guide* and the *Sybase Developer Edition Servers Installation Guide*.

You must also perform some procedures that are necessary for the tutorials to function properly.

Starting and exploring Sybase WorkSpace

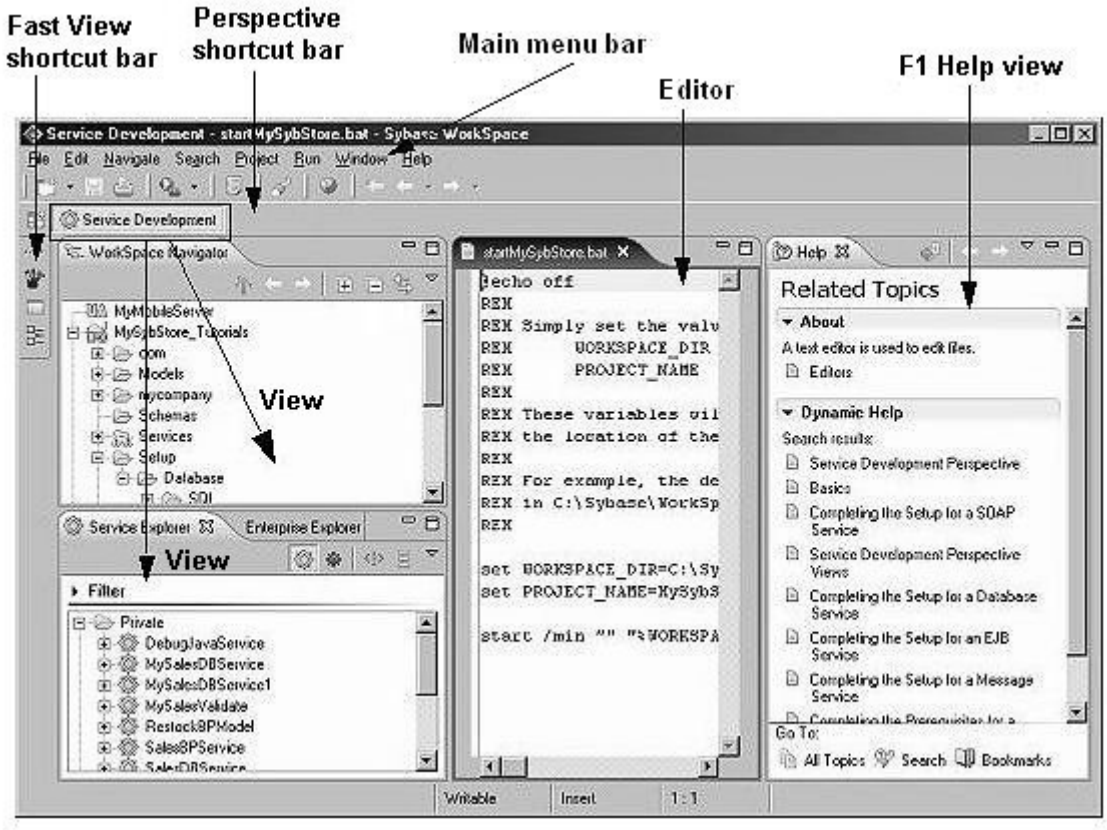
The Sybase WorkSpace main window is called the workbench. A workspace is the directory where your work is stored. When you initially start the application, you are prompted to select the workspace you want to use. Subsequently, you can choose a new workspace or specify that the program display the most recently used workspace.

To start WorkSpace, select **Start|Programs|Sybase|Sybase WorkSpace|Sybase WorkSpace 1.5**.

After you enter or select the workspace location, the WorkSpace main window opens, which displays one perspective. A perspective displays editors and views, such as the WorkSpace Navigator.

The following screen shows the WorkSpace main window (workbench) that displays the Service Development perspective, which includes the WorkSpace Navigator, the Service Explorer, and the Enterprise Explorer. Also open is the text editor and the F1 Help window.

On the far left is the Fast View toolbar, which contains icons for current views that are open but may be hidden. For example, to access the Tool Palette from the Fast View, right-click the Tool Palette title tab and select Fast View. You can also right-click the Tool Palette title tab, select Detached and move the Tool Palette view anywhere on your screen.



Note Depending on the activities you have performed previously or the perspectives and views that are already open, the views shown here may be located in a different area of the WorkSpace main window.

Becoming familiar with the Eclipse environment

If you are new to Eclipse, take some time to review the Eclipse online documentation on the Sybase WorkSpace bookshelf.

To access the Eclipse online documentation, select **Help**|**Help Contents** from the main menu bar in the Sybase WorkSpace window. When the **Help** window opens, select the *Workbench User Guide* in the **Contents** pane to learn more about Eclipse.

Setting up the SybStore tutorial

To prepare your WorkSpace installation to run the Service Development and Process Orchestration tutorials, you must complete these steps, in this order:

1 Download the MySybStore_Tutorials project.

2 Import the tutorial files into Sybase WorkSpace.

3 Start and connect to the MySybStore database.

4 Initialize the tutorial database.

5 Start and connect to the Unwired Orchestrator server.

6 Start and connect to the Unwired Accelerator server.

Downloading the MySybStore_Tutorials project

Before you begin the tutorials, download and import the files that create the MySybStore_Tutorials project, which contains all of the resources you need.

1 In an Internet Web browser, go to the Sybase Web site at http://www.sybase.com/ and click **Downloads**.

2 On the **Downloads** page, click **CodeXchange**. You see the Login page where you log in or create a MySybase account.

3 If you have a MySybase account, enter your **E-mail Address** and **Password**, click **Login**, and go to step 4.

If you do not have a MySybase account, click **Register now!** and follow the steps to create an account.

Once you are logged in, you see the CodeXchange registration page.

Note Although there are user e-mail and password login fields, you do not have to log in here or create another account.

4 Select the **Community** tab, and beneath "**Specific Product-related Projects Include**," select **Sybase WorkSpace**.

5 When the **Project Home** page opens, scroll to the **Popular Folders** table and click **SybStore** beneath **Tutorials v1.5**.

You see the **WorkSpace Documents & Files** page for the SybStore tutorials.

6 Right-click **MySybStore Tutorials Project Zip** and select **Save Target As** from the context menu.

7 When the **File Download** dialog box displays, navigate to the location where you want to save the file and click **Open**.

8 You see a progress bar indicating that the file is downloading to the selected location.

Importing the tutorial files into Sybase WorkSpace

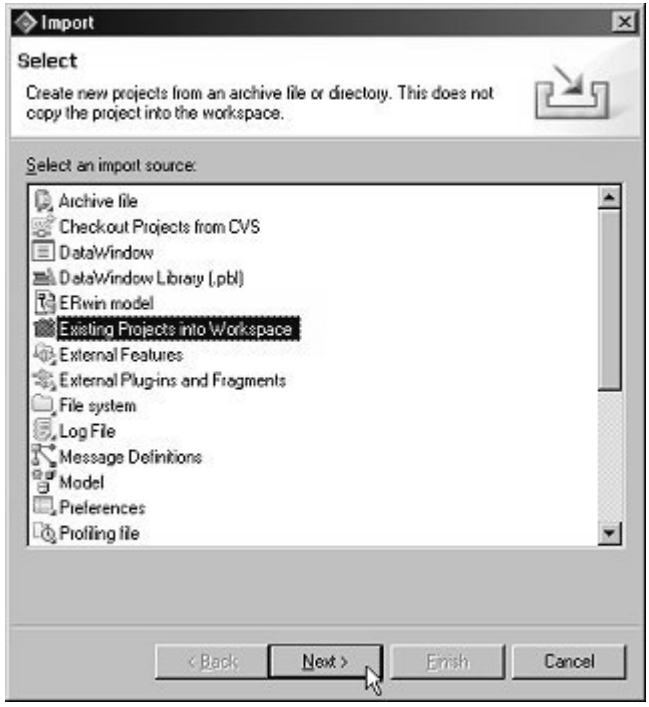
1 If WorkSpace is not running, select **Start|Programs|Sybase|Sybase WorkSpace|Sybase WorkSpace 1.5**.

2 If the Sybase WorkSpace **Welcome** page is open, select **Window|Close All Perspectives** to start with a blank window.

3 To open the Service Development perspective, select **Window|Open Perspective|Other**, choose **Service Development (default)** from the **Select Perspective** dialog box, and click **OK**.

4 Select **File|Import** from the WorkSpace main menu bar.

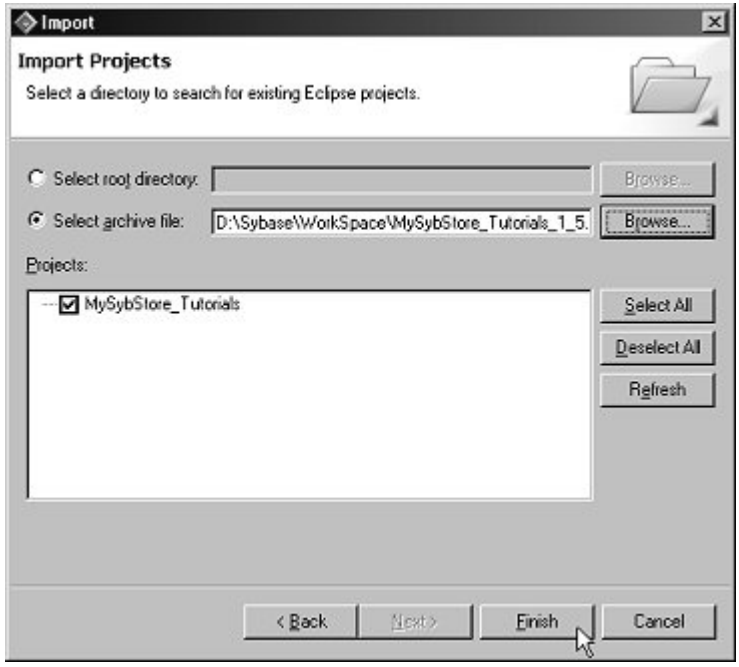
5 When the **Import** window opens, select **Existing Projects into WorkSpace** and click **Next**.



6 When the **Import Projects** dialog box opens, choose the **Select Archive File** option and click **Browse.**

7 When the file selection window opens, navigate to the **MySysStore_Tutorials_1.5.zip** file and click **Open**.

8 In the **Projects** list, verify that **MySybStore_Tutorials** is selected and click **Finish**.



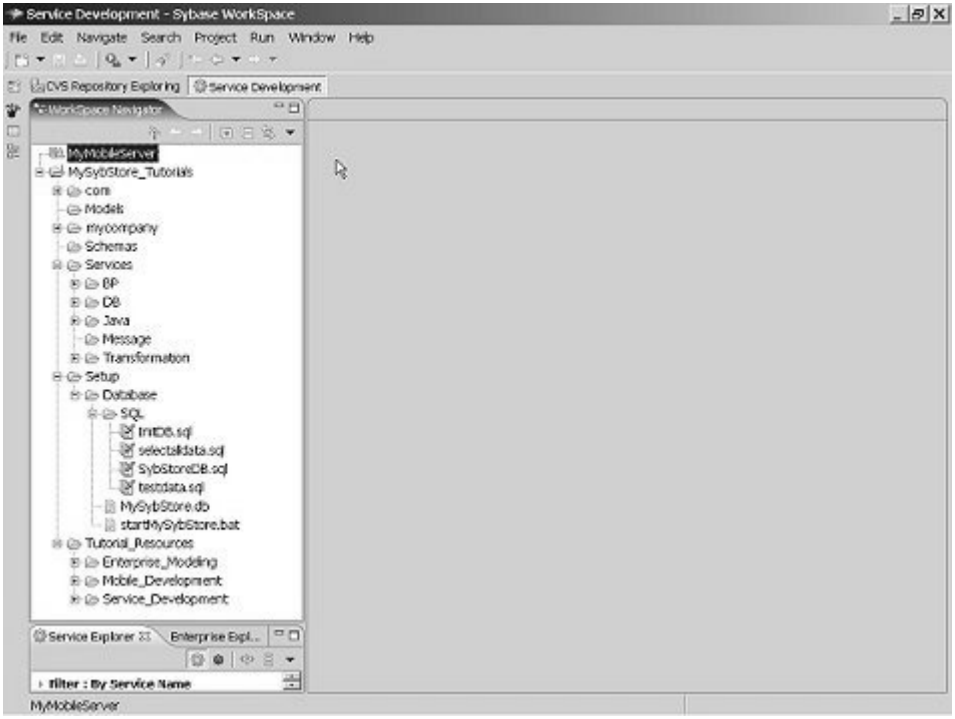
WorkSpace imports the project, which now displays in the WorkSpace Navigator window.

Note If the MySybStore_Tutorial project displays red "X"s on the project icon in the WorkSpace Navigator, right-click the project name and select **Update WorkSpace Build Path Entries** from the context menu.

When you see a message stating that WorkSpace has updated the project's build path entries, click **OK**. The red "X"s clear.

Exploring the MySybStore_Tutorial project

All of the resources you need or create for the tutorials are stored in the MySybStore_Tutorials tutorial project. To view the project's initial contents, expand the MySybStore_Tutorials project in the WorkSpace Navigator.



These are the top-level folders in MySybStore_Tutorials:

Folder	Description
com	Automatically generated based on package names contained in services and schemas.
Models	Location to which you should save tutorial models.
mycompany	Location in which the database service proxy files are stored when they are generated.
Services	Folder to which service files should be saved. This folder contains subfolders for each service type, such as BP (Business Development), DB (Database), Java, Message, and Transformation, to help categorize the services.
Setup	Contains the tutorial database and SQL scripts. Use the files in this folder to re-create the original database.

Folder	Description
Tutorial_Resources	Contains examples of files that you create in various tutorials and miscellaneous resources, such as XSD files, required by the tutorials.

As you complete each lesson, you will use or modify existing resources and add new resources to the project.

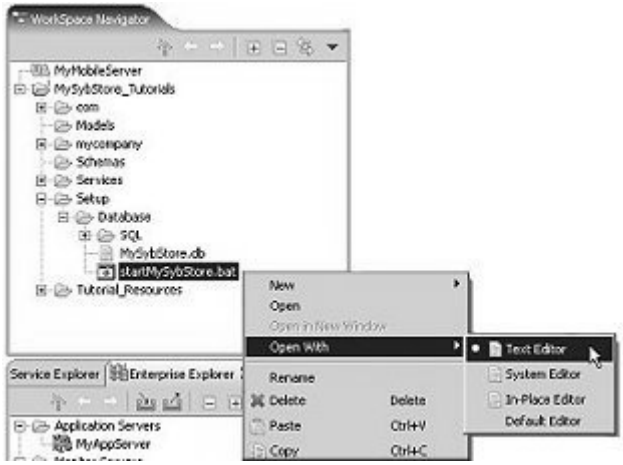
Starting and connecting to the *MySybStore* database

The MySybStore database is a Sybase Adaptive Server Anywhere database, located in the MySybStore_Tutorials tutorial project you created. Because the tutorial database is required, you must start and connect to the MySybStore tutorial database.

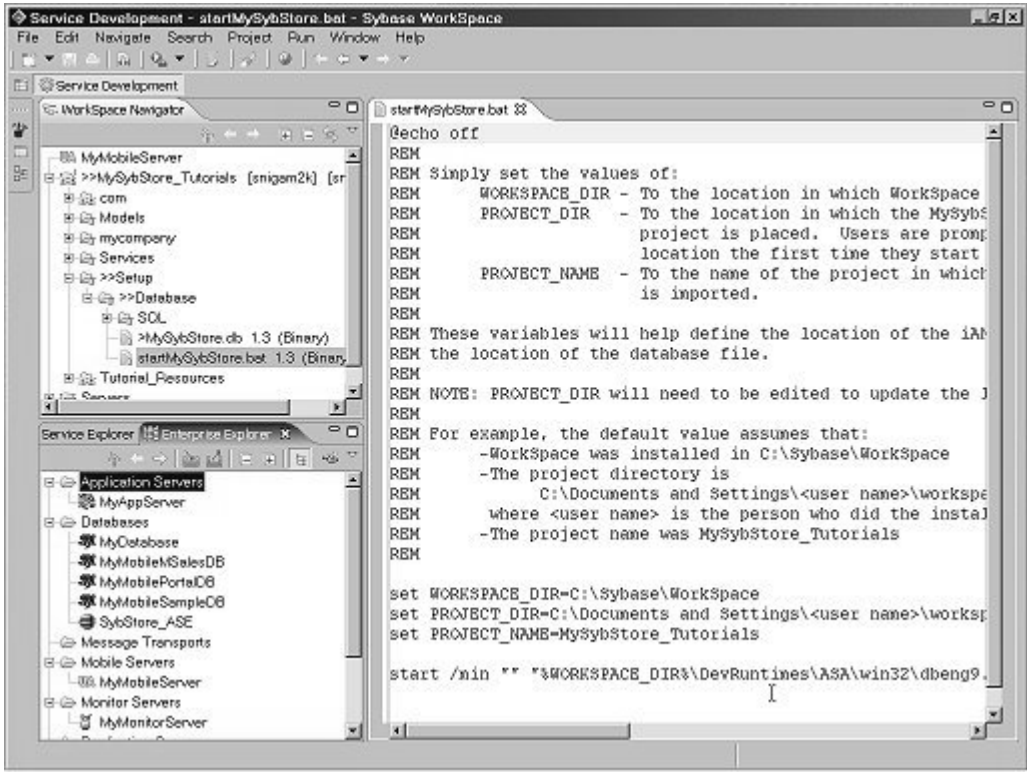
Starting the tutorial database

1 In the **WorkSpace Navigator**, expand the **MySybStore_Tutorials/Setup/Database** folder.

2 In the **WorkSpace Navigator**, right-click **startMySybStore.bat**, and select **Open With|Text Editor** from the context menu.



The batch file opens in the text editor.



3 Edit these lines in *startMySybStore.bat* to set the appropriate values for your installation and project:

```
set WORKSPACE_DIR=C:\Sybase\WorkSpace
set PROJECT_DIR=C:\Documents and Setting\<username>\workspace
```

- *WORKSPACE_DIR* should point to the directory in which Sybase WorkSpace is installed; for example *D:\Sybase\WorkSpace*.
- *PROJECT_DIR* should point to the directory where your project files are stored. The default is *C:\Documents and Settings\<username>\workspace*, which is set when you first start WorkSpace. If you selected a different location for your personal workspace, change this variable's value to point to that location.

  For example, if you created your personal Sybase WorkSpace on *D:\Sybase\<username>\workspace* when you installed the product, change this value to point to that location.
- Verify that the port number specified on the last line is `2658`

4 Select **File|Save** from the main menu bar to save the changes.

5 Select **File|Close** from the main menu bar to close the editor.

6 To start the database, right-click **startMySybStore.bat** and select **Open With|System Editor** from the context menu.

The Adaptive Server Anywhere, Developer Edition window appears for a few seconds, then you see the Adaptive Server Anywhere icon in your Windows' system tray, indicating that the database is running.

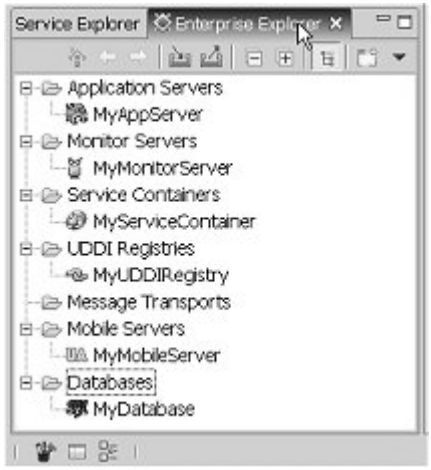
Creating a database connection profile

A connection profile must exist for the MySybStore tutorial database, which allows WorkSpace to connect to the database once the database has been started.

Note The first time you execute a SQL file, you must specify the connection profile. Subsequently, you do not have to specify the connection profile unless you want to change it. You can assign different connection profiles to different files, which allows you to use different ports or user names and passwords.

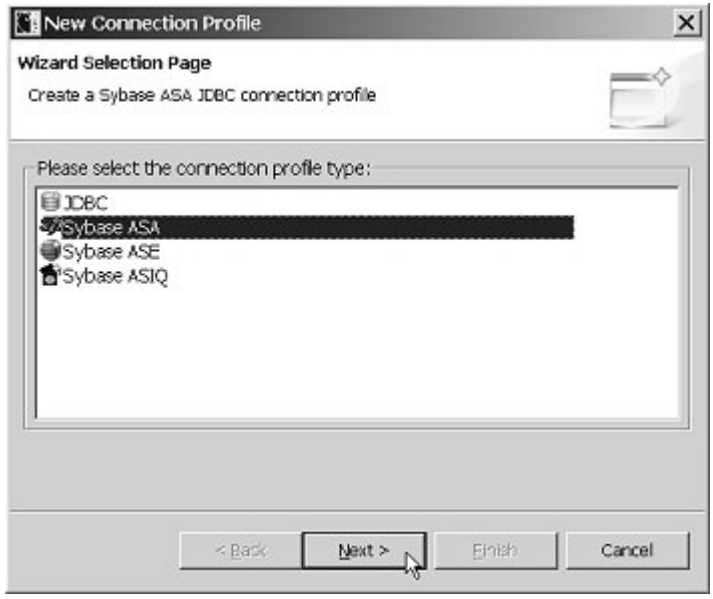
A connection profile contains the connection information (for example, host name and port) that WorkSpace uses to connect to a server resource. Create and configure connection profiles in the Enterprise Explorer.

1 Select the **Enterprise Explorer** tab if that view is open. If that view is not open, select **Window|Show View|Enterprise Explorer** to open the view.

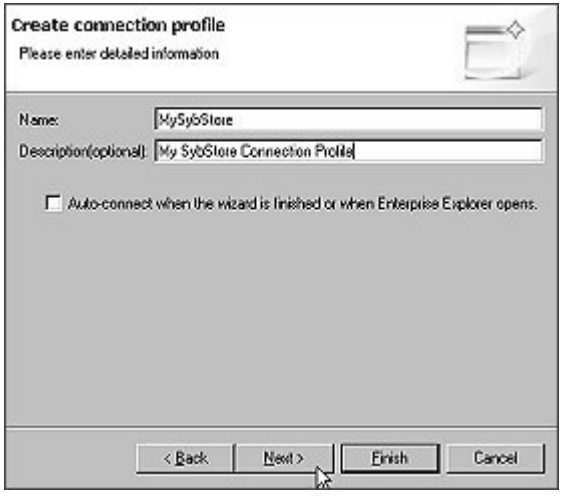


2 To create a new database connection profile, in the **Enterprise Explorer**, right-click **Databases** and select **New** from the context menu.

3 When the **New Connection Profile** wizard appears, select **Sybase ASA** from the connection profile type list and click **Next**.



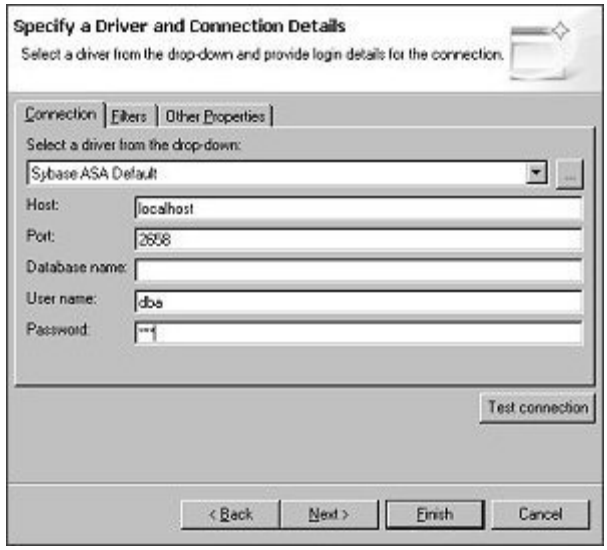
4 Enter `MySybStore` in the **Name** field, enter `MySybStore Connection Profile` in the **Description** field, then click **Next**.



5 On the **Driver and Connection Details** page of the wizard:

- Verify that **Port** is set to `2658`.

- Change **Password** to `SQL`.



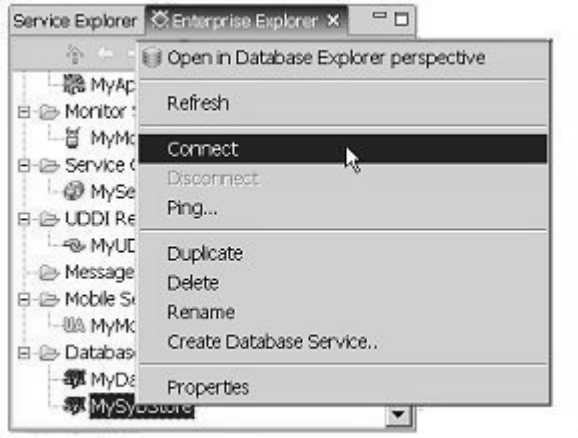
6 Select **Test Connection** to ensure the values you entered are correct.

7 When a prompt indicates that the `Ping Succeeded`, click **OK**.

If the ping fails, verify that the MySybStore database is running and that the values entered for the **Driver and Connection Details** are correct, then repeat step 6.

8 Click **Finish** to complete the connection profile.

9 In the **Enterprise Explorer** view, right-click the **MySybStore** connection profile in the **Databases** folder and select **Connect** from the context menu.



A successful connection is indicated when the database version appears beside the database name in the Enterprise Explorer and a database icon displays below the profile.

You have created a connection profile and connected to a running database. The MySybStore database is used in many of the Sybase WorkSpace tutorials.

Initializing the tutorial database

1 Complete the previous procedures in this section. WorkSpace must be running, the SybStore tutorial files must be installed, the SybStore database must be running, and you must be connected to the database in WorkSpace using the SybStore connection profile.

2 Select **Window|Open Perspective|Other**, choose **Service Development (default)** from the **Select Perspective** dialog box, and click **OK**.

3 Execute the *InitDB.sql* script:

a In the **WorkSpace Navigator**, expand the folder **MySybStore_Tutorials/Setup/Database/SQL,** right-click the *InitDB.sql* file, and select **Execute SQL File** from the context menu.

b When the **Select Profile for the Editor** dialog box opens, enter these values in the dialog box:

- Database Type – select **Adaptive Server Anywhere_9.x**.

- Connection Profile Name – select **MySybStore**.

c Click **OK**.

You see a progress window that indicates that the script is executing. When the script is finished running, you see the SQL Results view in the WorkSpace window.

4 Execute the *testdata.sql* script:

a In the **WorkSpace Navigator**, expand the folder **MySybStore_Tutorials/Setup/Database/SQL,** right-click the **testdata.sql** file, and select **Execute SQL File** from the context menu.

b When the **Select Profile for the Editor** dialog box opens, enter these values in the dialog box:

- Database Type – select **Adaptive Server Anywhere_9.x**.
- Connection Profile Name – select **MySybStore**.

c Click **OK**.

5 Click the "**X**" on the **SQL Results** window's title tab to close that view.

Starting and connecting to the Unwired Orchestrator server

Some of the tutorials in this guide require that the Unwired Orchestrator server be running and that a connection to that server be established in Sybase WorkSpace.

Starting the Unwired Orchestrator server

Start the Unwired Orchestrator server outside of Sybase WorkSpace using the Windows Start menu

1 Select **Start|Programs|Sybase|Sybase WorkSpace|UO 5.1|Start UO**.

EAServer starts and a command window appears. The UO51Runtime Adaptive Server Anywhere database also starts and the database icon appears in the Windows system tray.

2 Minimize (but do not close) any open command windows.

Connecting to the Unwired Orchestrator server

The Sybase WorkSpace sample uses the default Unwired Orchestrator connection profile, MyServiceContainer.

1 In the WorkSpace main window, select the **Enterprise Explorer** tab if that view is open. If that view is not open, select **Window|Show View|Enterprise Explorer** to open the view.

2 Expand the **Service Containers** folder to locate the **MyServiceContainer** default connection profile.

3 Right-click **MyServiceContainer** and select **Ping** from the context menu.

4 When the `Ping Succeeded` message displays, click **OK**.

If the ping fails, verify that the Unwired Orchestrator server is running and that the values entered for **Unwired Orchestrator Server Connection Properties** are correct.

5 In the **Enterprise Explorer** view, right-click **MyServiceContainer** and select **Connect** from the context menu.

A Packages folder appears in the view under MyServiceContainer, indicating the connection is successful.

CHAPTER 2

Service Development Tutorials

The Service Development tutorials teach you how to use Sybase WorkSpace tools to create several types of services.

Topic	Page
Overview	21
Creating a database service	22
Creating a Java service	27
Creating a transformation service	43
Creating a message service	64
Using a generic JMS provider for messaging	72
Packaging, deploying, and testing a service	79

Overview

Sybase WorkSpace provides tools for creating, discovering, editing, sharing, deploying, and publishing services.

You develop and manage services as discrete units, which are well defined, self-contained, and independent of the context or state of other services. You can publish services either to the Web or internally so that the business logic and data can be shared. You can also reuse services without altering them. For internally published services, you can derive a new service by using parts of the published service and adding new logic or data.

The Service Development tutorials teach you now how to create, package, deploy, and test various types of services.

For more information about WorkSpace Service Development, see the WorkSpace online help topic *Sybase WorkSpace Development*|*Service Development.*

Prerequisites

Before you begin the tutorials, complete all of the procedures in Chapter 1, “Introduction, Installation, and Setup.”

Creating a database service

A database operation that performs as a standalone service or performs as a service within a business process or other composite service must use a database service. You can use existing stored procedures, SQL statements created during development of the database service, and query files to perform a selected operation on a particular database. This results in a service file that you can deploy to provide access to the selected operation at runtime.

This tutorial teaches you how to create a database service using Sybase WorkSpace tools. After you complete this tutorial, you will know how to create a database service from a set of stored procedures. You will have a complete service, ready to be deployed and tested.

This tutorial contains one lesson.

Lesson 1: Creating a database service

In this lesson, you create a basic database service for the SybStore application.

1 To start Sybase WorkSpace, select **Start|Programs|Sybase|Sybase WorkSpace|Sybase WorkSpace 1.5**.

2 To open the Service Development perspective, select **Window|Open Perspective|Other**, choose **Service Development (default)** from the **Select Perspective** dialog box, and click **OK**.

   The Service Development perspective includes the WorkSpace Navigator view, the Service Explorer view, and Enterprise Explorer view in the WorkSpace main window.

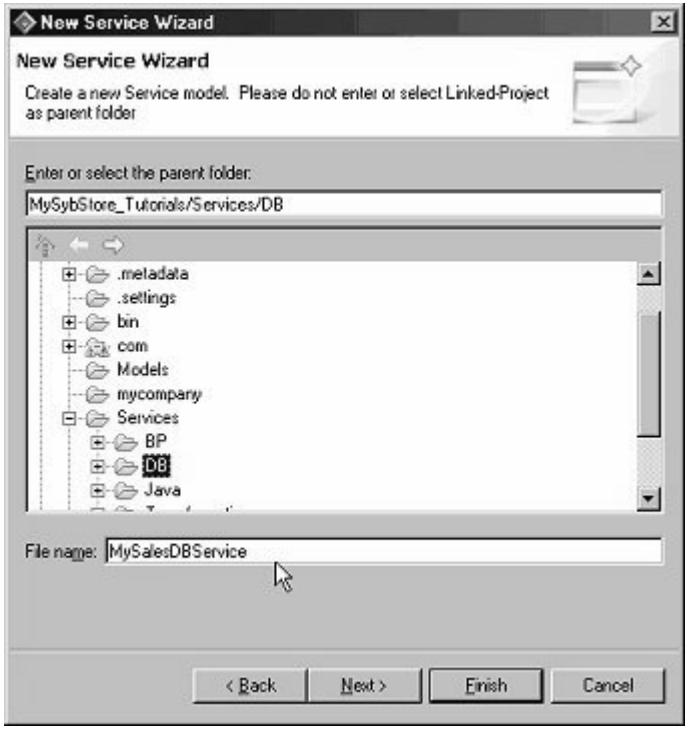
3 In the **Enterprise Explorer**, expand the **Databases** folder, right-click the **MySybStore** connection profile, and select **Create Database Service** from the context menu.

Note You can also create a database service by selecting **File|New|Service** from the WorkSpace main menu bar.

---

4 When the **New Service Wizard** opens, expand the **MySybStore_Tutorials/Services/DB** folder to populate the **Enter or Select the Parent Folder** field.

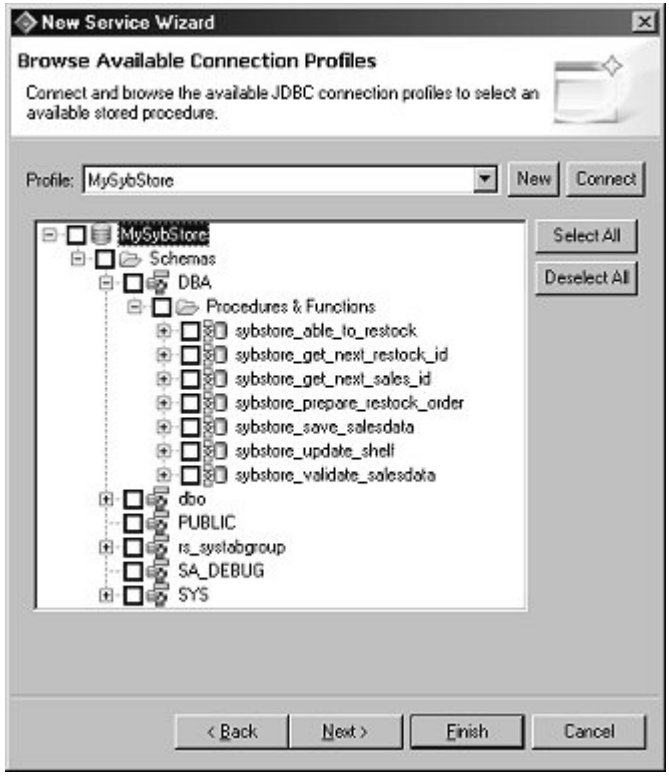
5 Enter `MySalesDBService` in the **File Name** field, then click **Next**.



6 On **Service Summary** page, click **Next**.

7 On the **Browse Available Connection Profiles** page, select **MySybStore** from the drop-down list for the connection **Profile**.

8 Expand **MySybstore/Schemas/DBA/Procedures & Functions**. You see the stored procedures in the SybStore tutorial database.



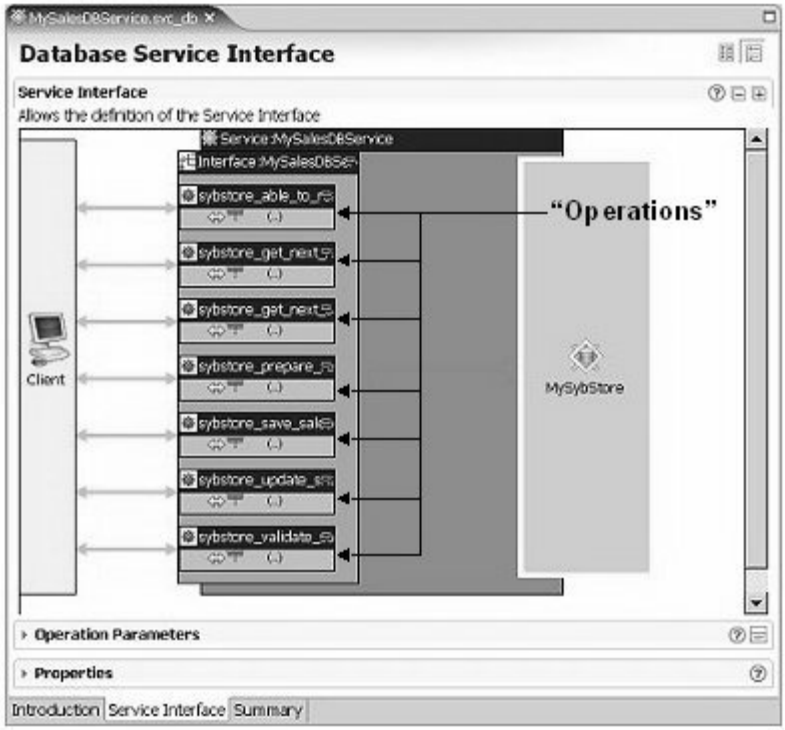
9 Select the **Procedures & Functions** check box, then click **Finish** to create a database service operation for each selected stored procedure in the MySybStore database.

The new database service is created (*MySalesDBService.Svc_db*) and opens in the Database Service Editor.

Note To enlarge the editor display, minimize or close other windows by clicking the "X" on the window's title tab. You can also click the editor's maximize button to it take up the entire WorkSpace window.

10 Select the **Service Interface** tab to see a graphical representation of the service's interface.

The diagram shows an operation for each stored procedure in the database.



11 In the **Service Interface** diagram, select the first operation within the **Interface:MySalesDBService** object—**sybstore_able_to_restock**.

12 Expand the **Properties** pane (at the bottom of the **Database Service Interface** tab) and select the **Autocommit** option.

13 For the other six operations, select the operation in the diagram, expand the **Properties** pane and select **Autocommit**.

14 Select **File|Save** from the WorkSpace main menu bar.

15 Select **File|Close** from the main menu to close the editor.

Creating a Java service

A Java service enables invocation of Java code within a business process by binding the Java service to a local Java class and the service operations to public methods within the Java class. Since a Java service can call other services through the Java service proxy interface, you can build composite services using a Java service to integrate process flow from many services into one service interface. You can deploy a Java services independently or incorporate them into a composite service.

This tutorial teaches you how to create a Java service using WorkSpace tools. After you complete this tutorial, you will know how to create a Java service, and you should understand the basic components of a Java service. You will have a complete Java service, ready to be deployed and tested.

When a Java service consumes other services, it is referred to as a composite Java service. Sybase WorkSpace automatically generates the necessary Java code to facilitate invocation of services from a Java service.

In the SybStore application, a Java service is used to validate incoming sales data.

This tutorial consists of:

Lesson 1: Creating a Java service
Lesson 2: Invoking a database service from Java service operation

Prequisites

This tutorial requires that you complete “Creating a database service” on page 22 first.

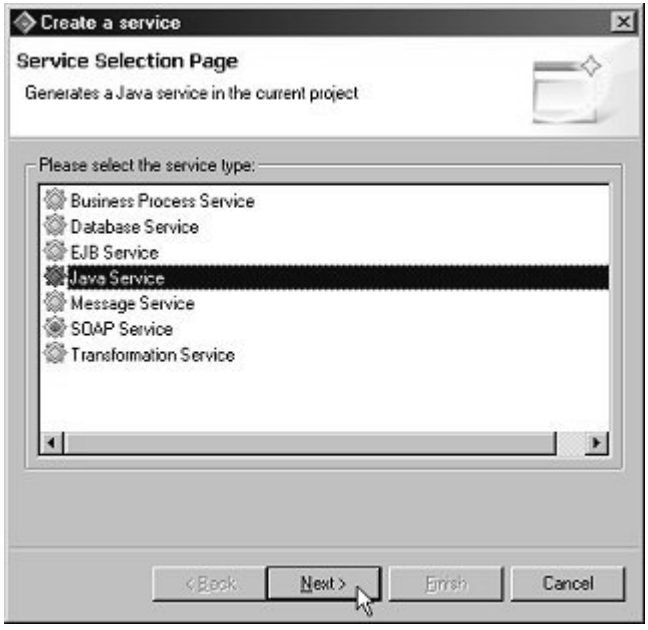
Lesson 1: Creating a Java service

In this lesson, you create a basic Java service for the SybStore application.

1 To open the **Service Development** perspective, select **Window|Open Perspective|Other**, choose **Service Development (default)** from the **Select Perspective** dialog box, and click **OK**.

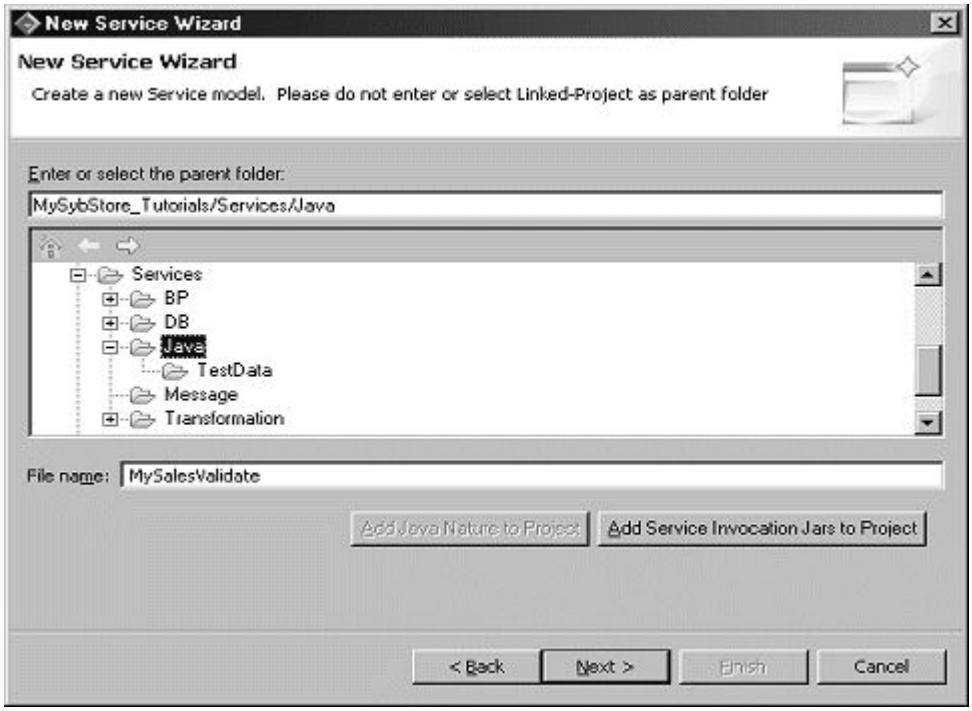
2 Select **File|New|Service** from the WorkSpace main menu bar. The **Create a Service** wizard appears.

3 On the **Service Selection Page**, select **Java Service** and click **Next**.

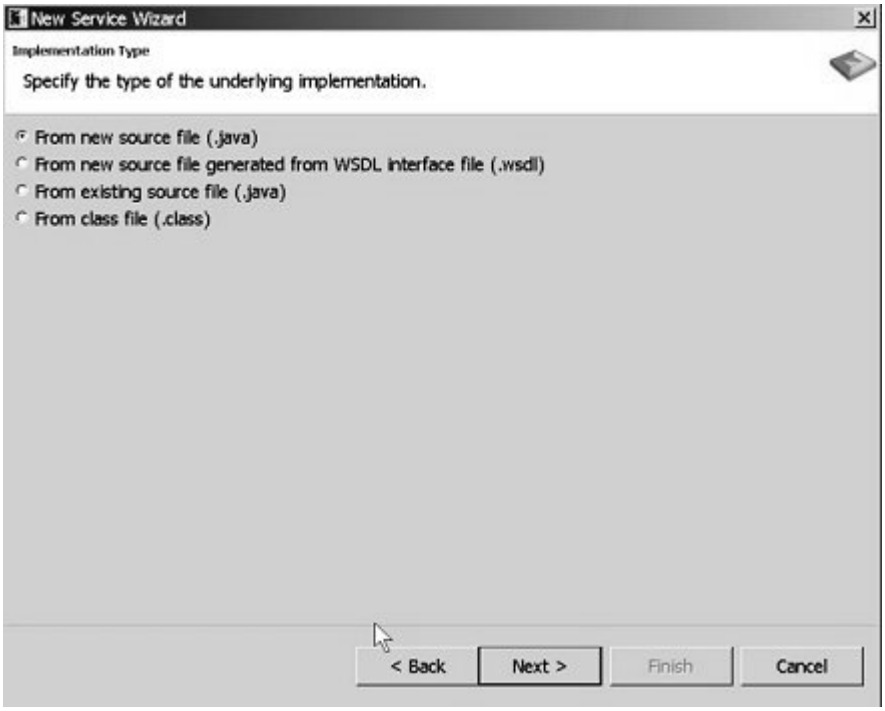


4 When the **New Service Wizard** page opens, select the **MySybStore_Tutorials/Services/Java** folder to populate the **Enter Or Select the Parent Folder** field.

5 Enter `MySalesValidate` in the **File Name** field and click **Next**.

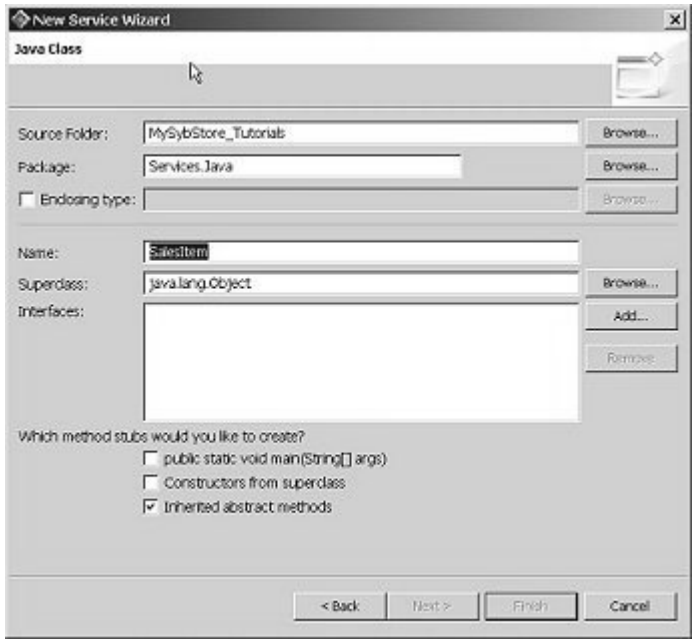


6 When the **Service Summary** window opens, click **Next**. The **Implementation Type** page appears.



7 Select the **From New Source File (.java)** option and click **Next**.

The **Java Class** window opens.



8 Enter `Services.Java` in the **Package** field, or click **Browse** and select **Services.Java** from the **Package Selection** window.

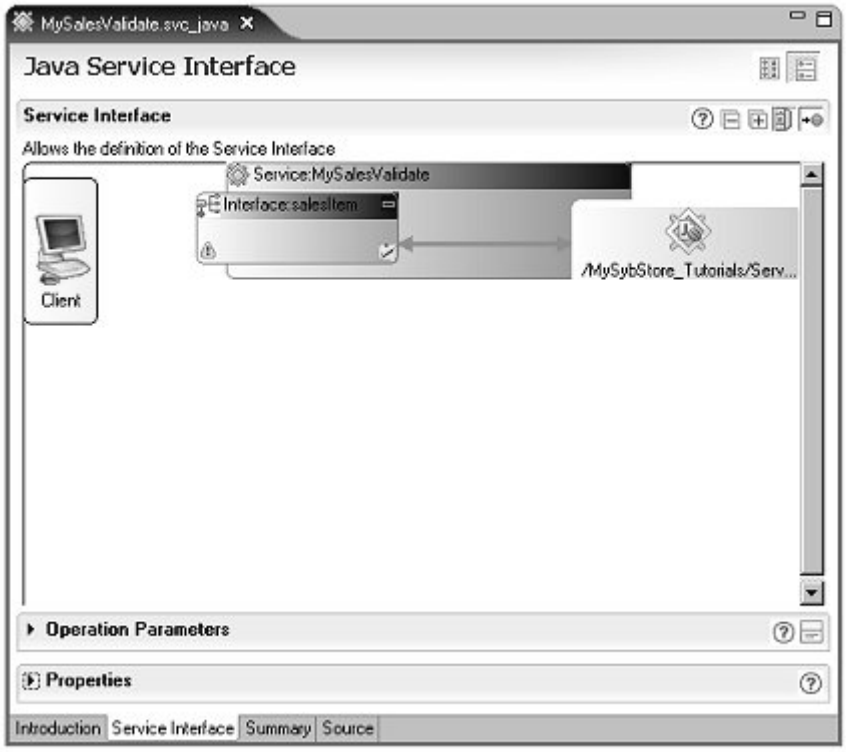
9 Enter `salesItem` in the **Name** field and click **Next**.

Note If you see a warning about the format of the package name, you can ignore it.

10 When the **Dependencies** page appears, click **Next**.

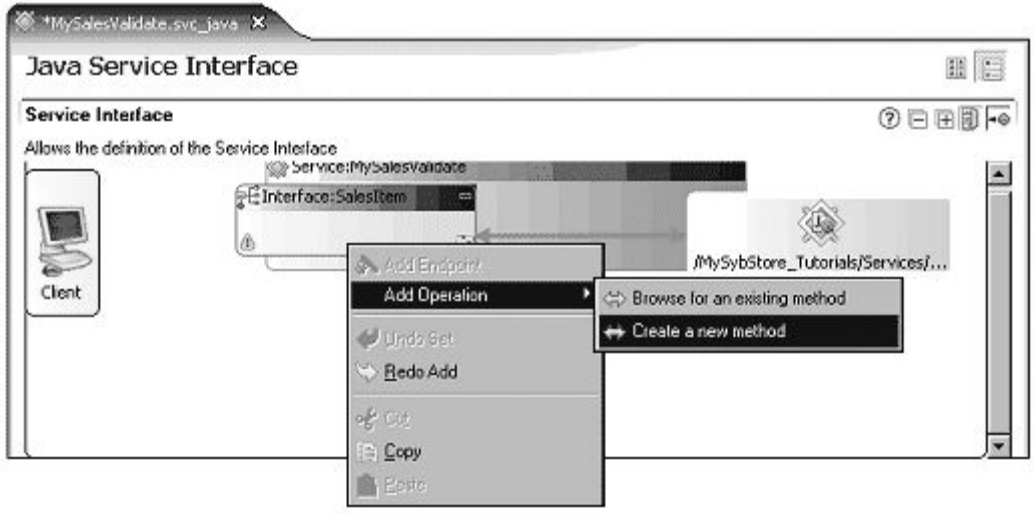
11 When the **Summary** page appears, click **Finish** to create the Java service. The service is created and opens in the Java Service Editor.

12 Select the **Service Interface** tab.



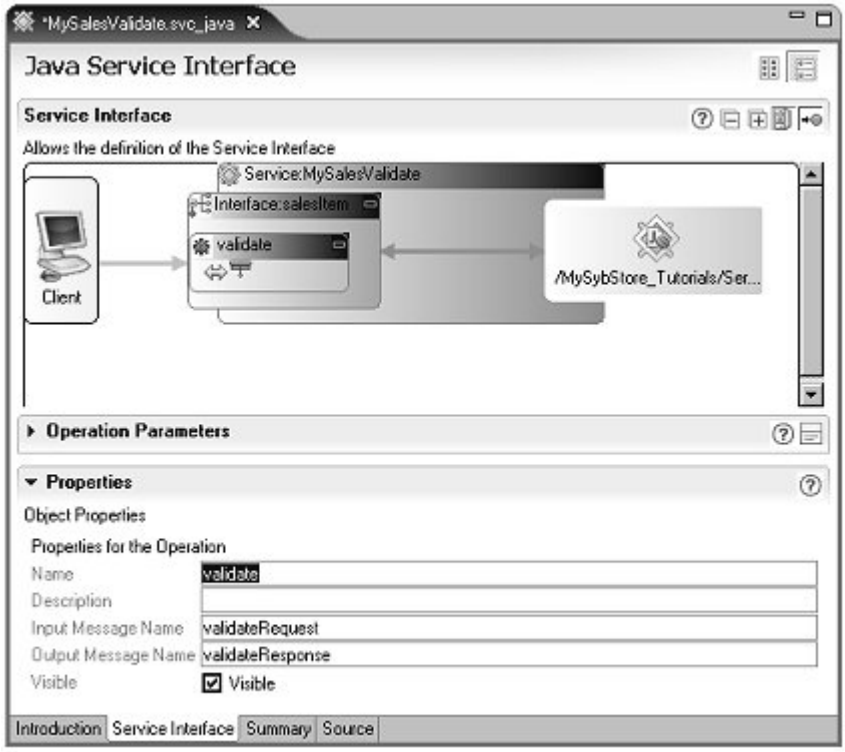
The Java Service Interface page contains three panes: a Service Interface diagram, Operation Parameters, and Properties.

13 In the **Service Interface** diagram, right-click the **Interface:SalesItem** box, and select **Add Operation|Create a New Method** from the context menu.



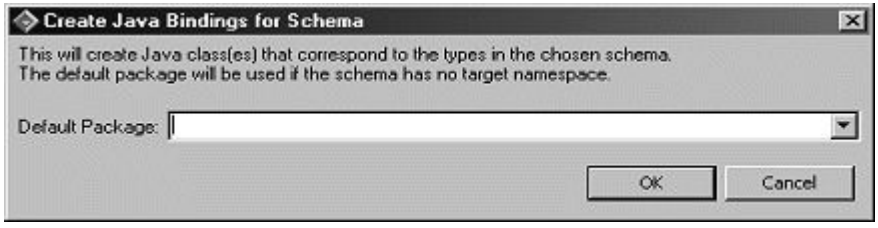
You see the new operation represented in the diagram.

14 Expand the **Properties** pane, select **operation1** in the diagram, and enter `validate` in the Object Properties **Name** field.



15 In the **WorkSpace Navigator**, expand the **MySybStore_Tutorials/Tutorial_Resources/Service_Development** folder.

16 Right-click the **SybStore.xsd** file and select **Create Java Bindings** from the context menu. The **Creat Java Bindings for Schema** dialog box displays.

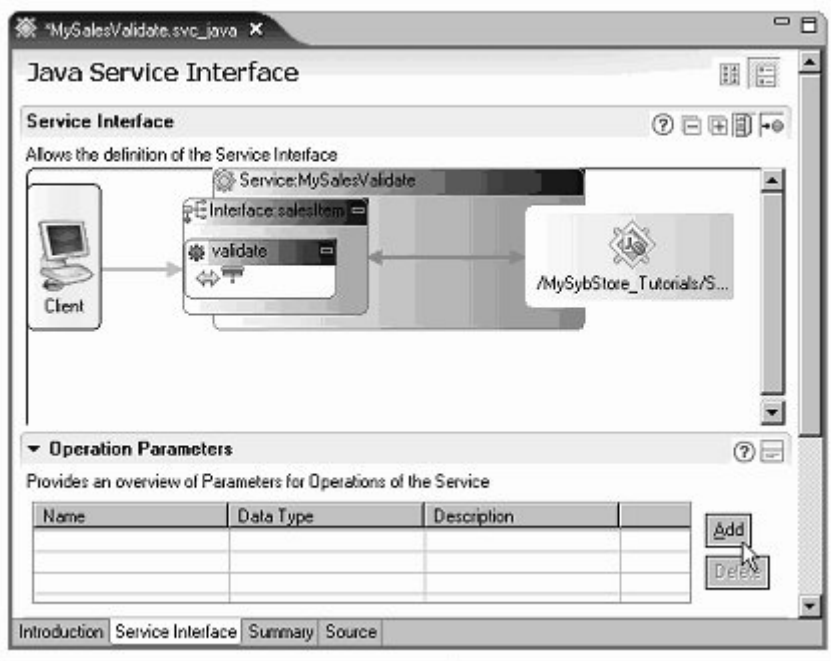


17 Leave the **Default Package** field empty and click **OK**. This generates Java types for the XSD schema, which provides the seamless ability to handle XSD-based type definitions and generated Java class types.

18 When you see the message "`Java Binding creation was successful`," click **OK**.

19 Open the **Operation Parameters** pane in the center of the Java Service Editor.

20 In the **Service Interface** pane, select the **validate** operation box in the diagram, then click **Add** in the **Operation Parameters** pane to add a new parameter.

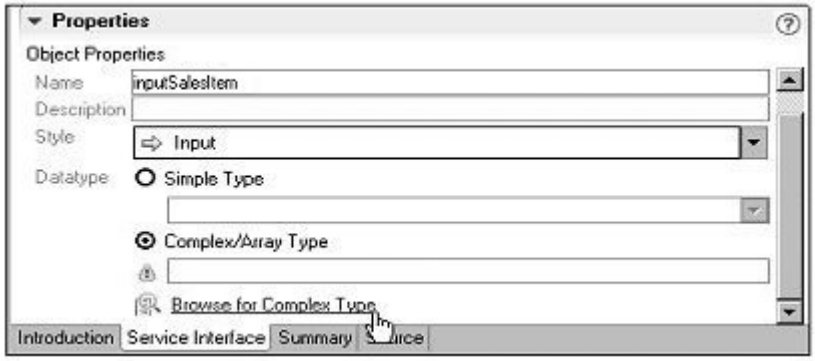


21 In the **Operation Parameters** pane, select the new parameter in the table, which by default is named **newParameter1**.

22 In the **Properties** pane (below the **Operation Parameters** pane), enter `inputSalesItem` in the **Name** field.

23 Continuing in the **Properties** pane, scroll down if necessary, and select **Complex/Array Type** for the **Datatype**.

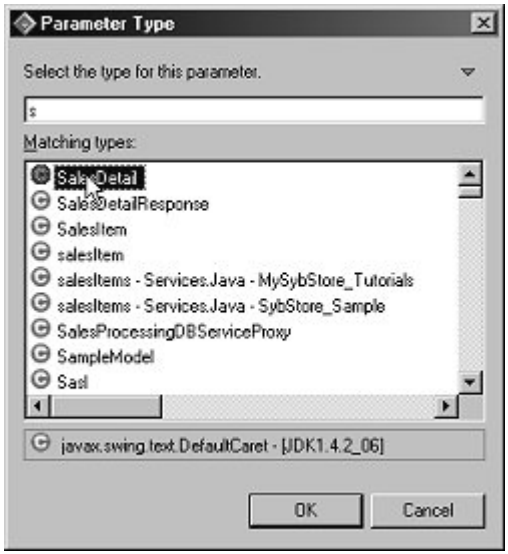
24 Click **Browse for Complex Type** at the very bottom of the **Properties** pane.



25 When the **New Parameter** dialog box opens, click **Browse** for the **Type** field.

26 When the **Parameter Type** dialog box opens, enter the letter “s” in the **Select the Type For This Parameter** field.

Note You must enter the first letter of the parameter type you are searching for in the Select the Type For This Parameter field, or the Matching Types list will remain empty.



27 Scroll down the list of **Matching Types** and select **SalesDetail**. When you select **SalesDetail**, the selection changes to `SalesDetail - com.sybase.workspace.tutorials.sybstore.schemas`.

28 Click **OK**.

29 When you return to the **New Parameter** dialog box, click **OK**.

30 Select **File|Save** from the WorkSpace main menu.

Leave *MySalesValidate.svc_java* open in the editor for the next lesson.

Lesson 2: Invoking a database service from Java service operation

In this lesson, you add Java code to call a database service operation (the sybstore_validate_salesdata stored procedure) from the Java service validate method.

Before you do this lesson, complete “Creating a Java service” on page 27.

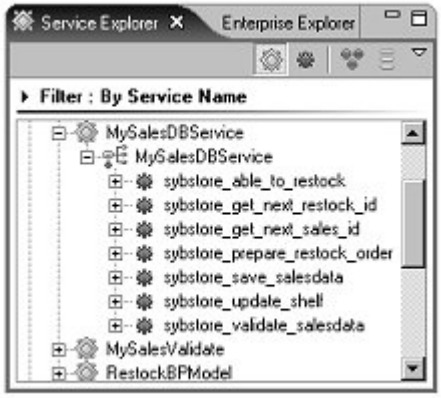
1 Open the Service Development perspective. Select **Window|Open Perspective|Other**, choose **Service Development (default)** from the **Select Perspective** dialog box, and click **OK**.

2 With **MySalesValidate.svc_java** open in the Java Service Editor, select the **Service Interface** tab.

3 Select the **Service Explorer**. This view may be hidden behind the **Enterprise Explorer**. If you do not see the **Service Explorer** tab, click **>>1** to the right of the **Enterprise Explorer** tab, then select **Service Explorer** from the list.

4 Expand the **Private** folder, which contains a list of services.

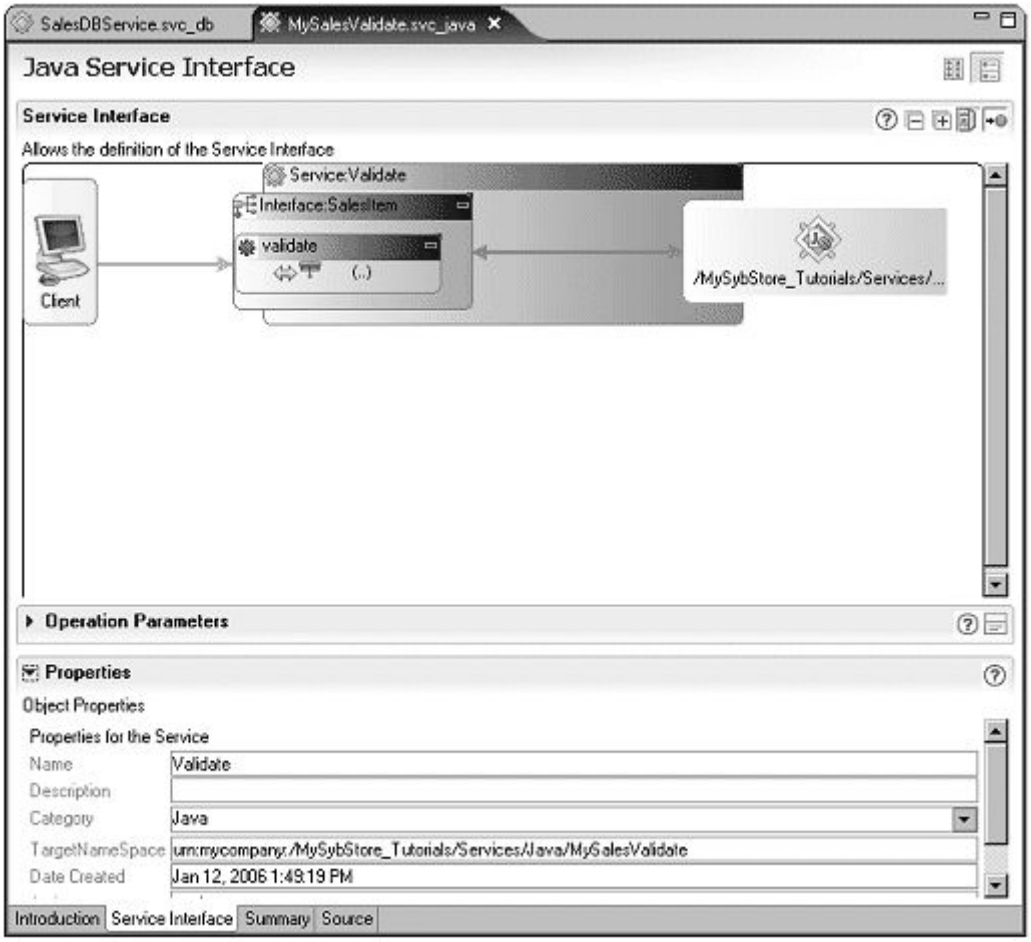
5 Expand **MySalesDBService/MySalesDBService** to display a list of services.



6 In the **Service Explorer**, double-click the **sybstore_validate_salesdata** operation at the end of the list.

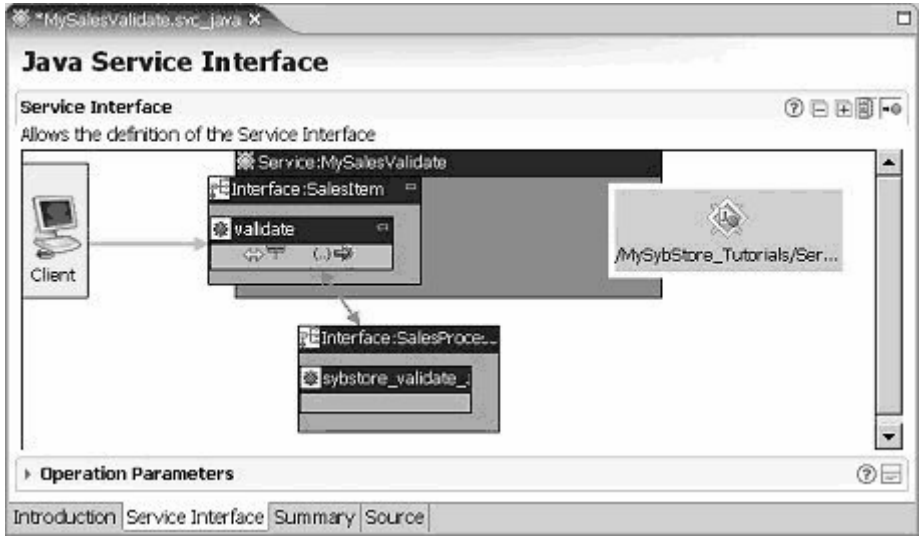
There are now two files in editors—one for the Java service (*MySalesValidate.svc_java*) and one for the database service that you created previously (*MySalesDBService.svc_db*).

7 Select the **MySalesValidate.svc_java** tab and verify that the **Service Interface** tab is selected.



8 In the **Service Explorer**, select the **sybstore_validate_salesdata** operation and drag and drop it onto the **validate** operation box in the **MySalesValidate** diagram.

9 When you see a message stating that the service proxy was generated successfully, click **OK**. The Java Service Interface diagram changes to show the call to the database service operation.



10 Select the **Source** tab for **MySalesValidate.svc_java**.

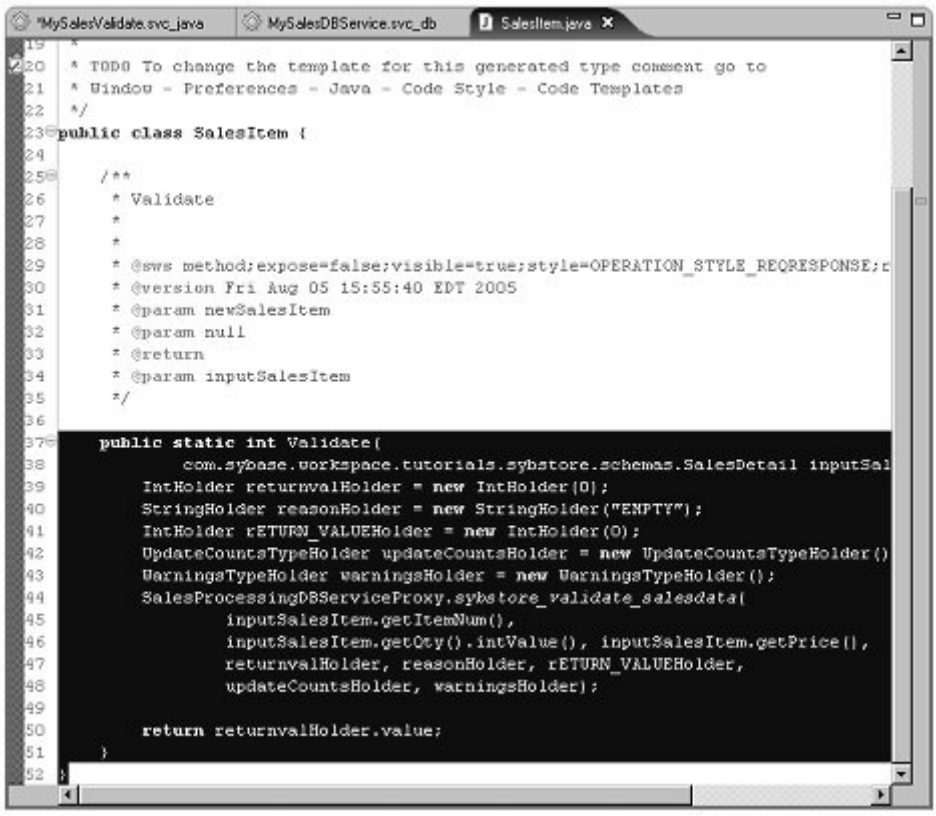
11 In the **WorkSpace Navigator**, expand **MySybStore_Tutorial/Tutorial_Resources/Service_Development/Java**, and double-click **SalesItem.java** to open the file in the Java editor.

12 Copy the validate method from the **MySybStore_Tutorial/Tutorial_Resources/Service_Development/Java/SalesItem.java** file

a Place your cursor at the beginning of the line that begins with:

```
public static int Validate(
```

b Hold down the cursor and drag to the end of the file to copy the entire method.



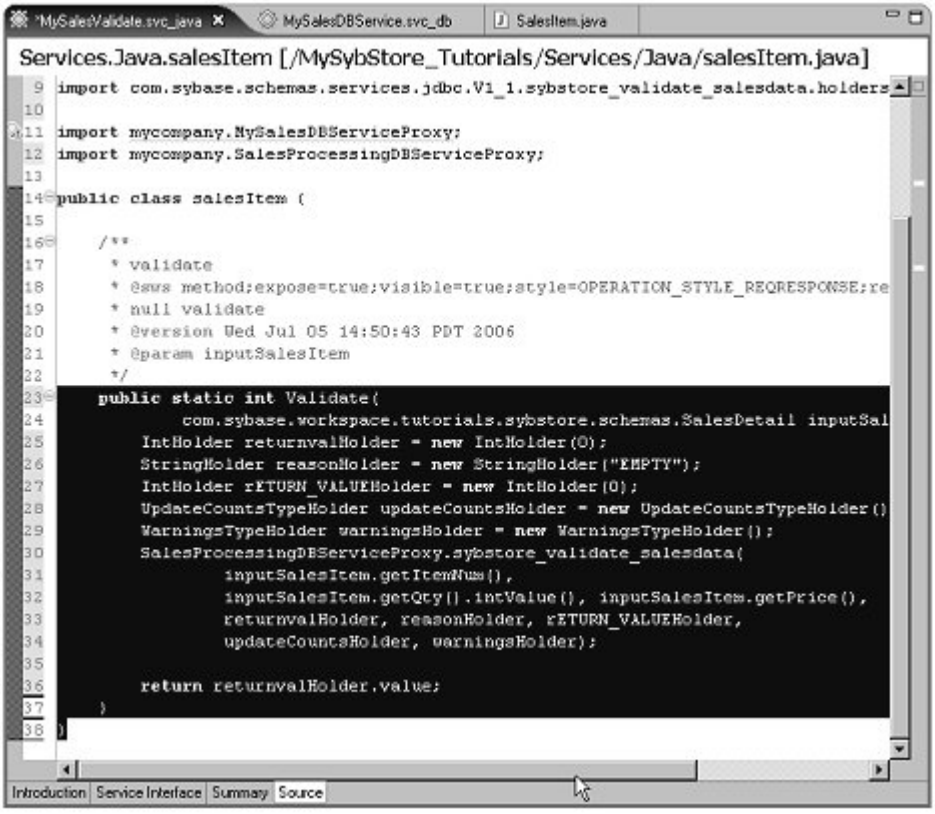
c Right-click in the editor and select **Copy** from the context menu.

13 Replace the validate method code in *MySalesValidate.svc_java* file by pasting in the Validate method code from the *SalesItem.java* file:

a Place your cursor at the beginning of the line that begins with:

```
public static void validate(
```

b Hold down the cursor and drag to the end of the file to copy the entire method.



c Right-click in the editor and select **Paste** from the context menu.

14 Select **File|Save** from the WorkSpace main menu bar to save **MySalesValidate.svc_java**.

15 Select the **Service Interface** tab for **MySalesValidate.svc_java** in the editor.

16 Expand the **Operation Parameters** pane and select the Validate operation in the diagram. Notice that the copied operation is "`Validate`" rather than "`validate`."

There is a second parameter named ValidateReturn. Because the code that you copied and pasted into this service contained a method return parameter, this output parameter was automatically added to the Validate operation.

17 Select **File|Close All** from the WorkSpace main menu to close all of the editor windows.

Note To learn how to deploy and test this service, follow the instructions in "Packaging, deploying, and testing a service" on page 79.

Creating a transformation service

A transformation service maps the content of an XML document from one XML schema to another. The result is a transformation service file you can use inside a business process service (Java transformation) or deploy to provide transformation functionality at runtime (XSL transformation).

This tutorial teaches you how to create a transformation service using Sybase WorkSpace tools. After you complete this tutorial, you will know how to create a transformation service, define its mapping logic, and test it. You will have a complete transformation service that can be used in other Sybase WorkSpace tutorials.

This tutorial consists of:

Lesson 1: Creating a new transformation service
Lesson 2: Defining mapping for a transformation service
Lesson 3: Testing a transformation service

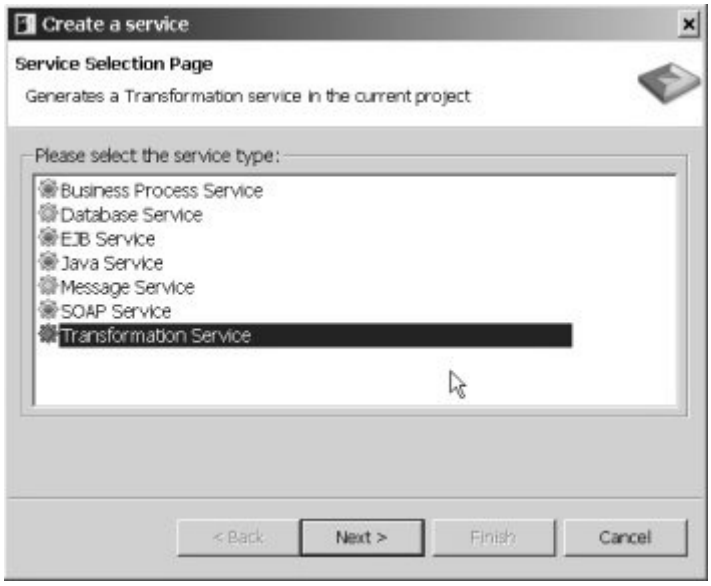
Lesson 1: Creating a new transformation service

In this lesson, you create a new transformation service for the SybStore application.

1 Open the **Service Development** perspective.

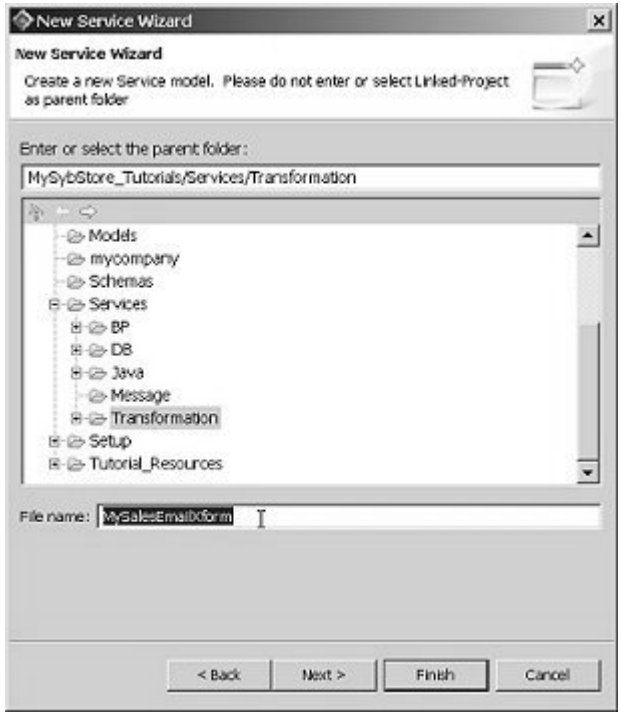
2 Select **File|New|Service** from the main menu bar.

3 In the **Service Selection Page** window, select **Transformation Service** and click **Next**.



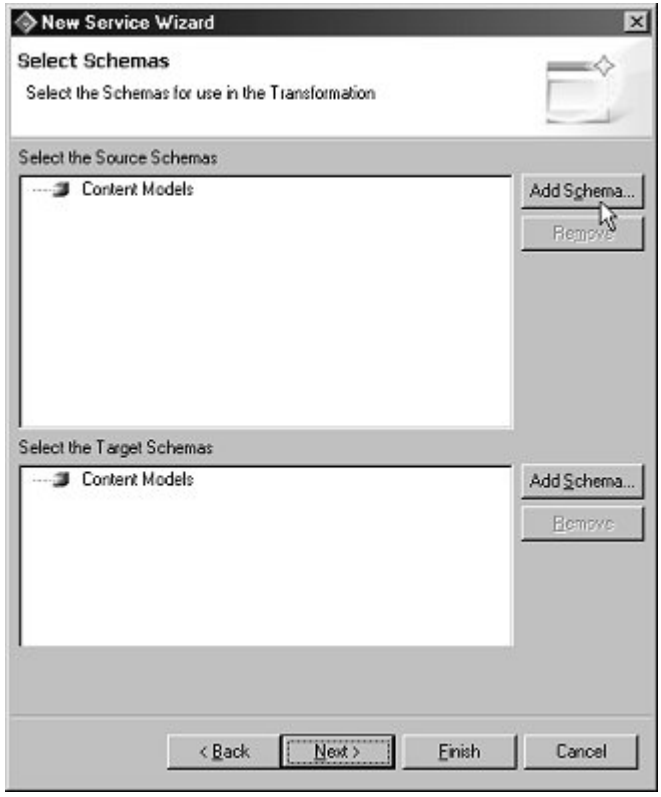
4 When the **New Service Wizard** window opens, expand **MySybStore_Tutorials/Services/Transformation**, the parent folder, to populate the **Enter or Select the Parent Folder** field.

5 Enter `MySalesEmailXform` in the **File Name** field and click **Next**.

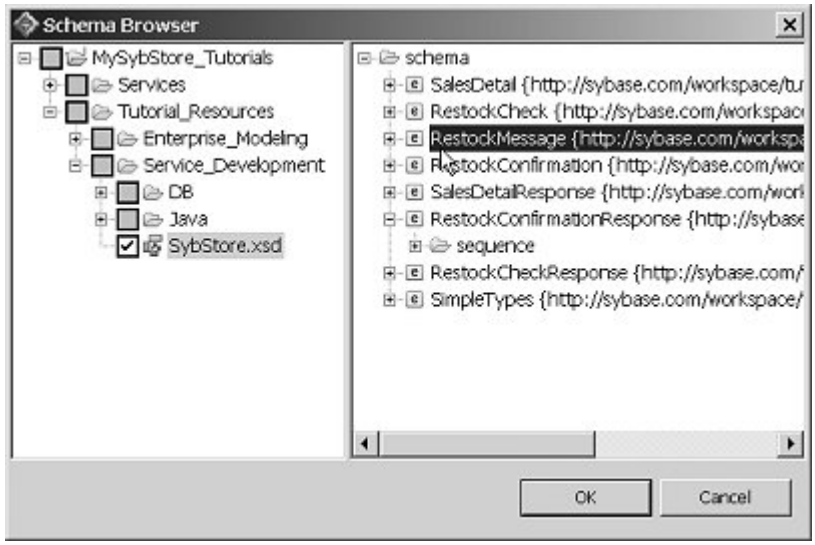


6 When the **Service Summary** page appears, select **Transformation** in the **Category** list box and click **Next**.

7 When the **Select Schemas** page appears, beside **Select the Source Schemas** pane, click **Add Schema**.



8 When the **Schema Browser** appears, expand the folders **MySybStore_Tutorials/Tutorial_Resources/Service_Development**, then locate and select the **SybStore.xsd** file.

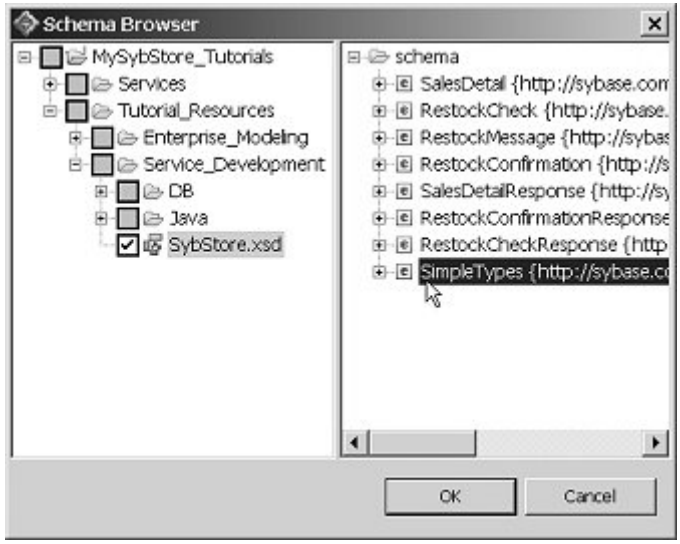


Elements in the XSD appear in the right pane of the Schema Browser.

9 Select the **RestockMessage** schema in the right pane and click **OK**.

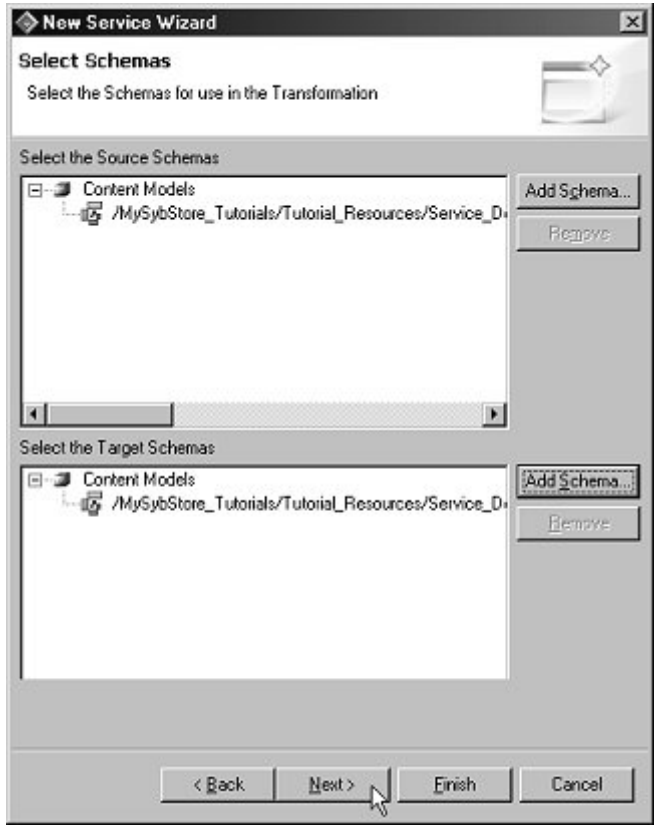
10 On the **Select Schemas** page, click **Add Schema** beside the **Select the Target Schemas** pane.

11 When the **Schema Browser** appears, expand the folders **MySybStore_Tutorials/Tutorial_Resources/Service_Development**, then locate and select the **SybStore.xsd** file.



12 Select the **SimpleTypes** schema in the right pane and click **OK**.

13 Click **Next** in the **Select Schemas** window.



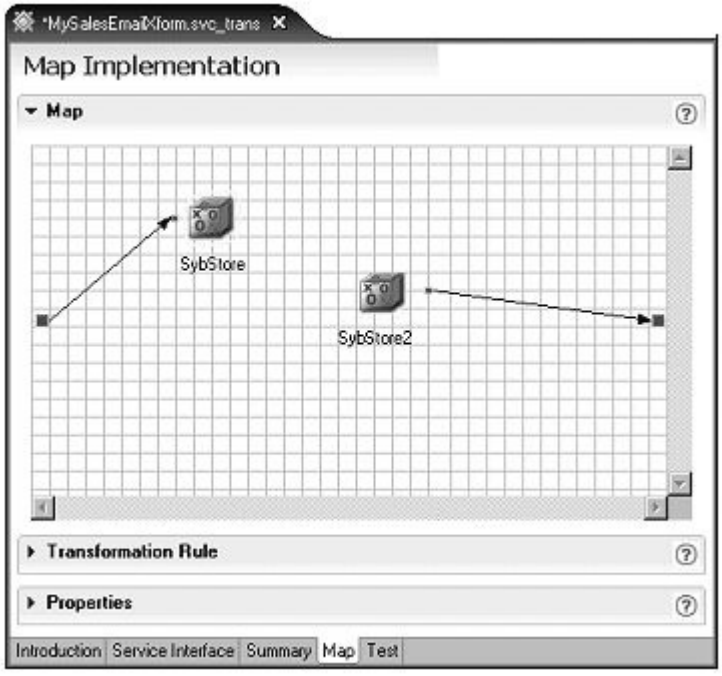
14 When the **New Service Wizard Summary** page appears, click **Finish**.

The service is created and opens in the Transformation Service Editor.

15 Select the **Map** tab at the bottom of the editor.

A diagram displays the unmapped source and target models. In the next lesson, you define the mapping.

16 Click the minimize button on the boxes in the diagram to view them as icons.



17 Select **File|Save** from the WorkSpace main menu.

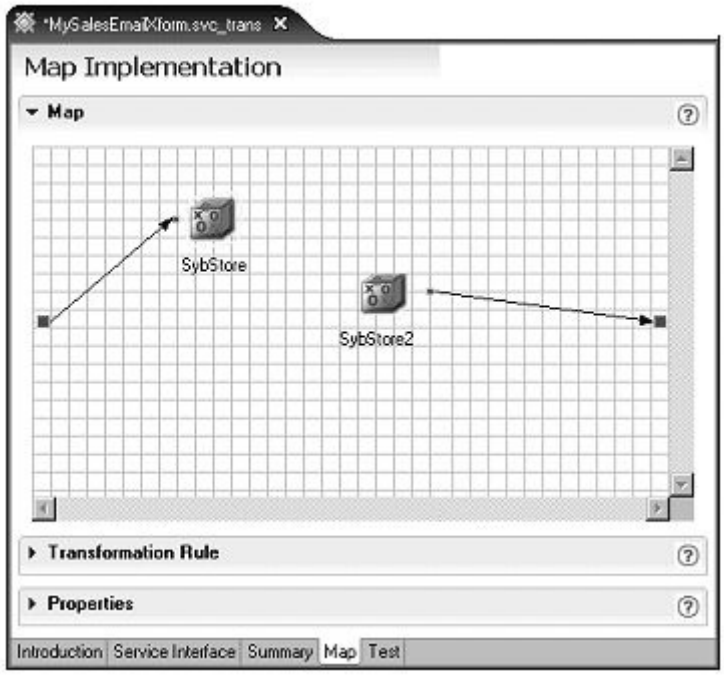
Leave the new transformation service file open for the next lesson.

Lesson 2: Defining mapping for a transformation service

In this lesson, you define the transformation mapping from the source content model to the destination content model, as identified by the XML schemas you chose in the preceding lesson.

You must complete “Lesson 1: Creating a new transformation service” on page 43 before you begin this lesson.

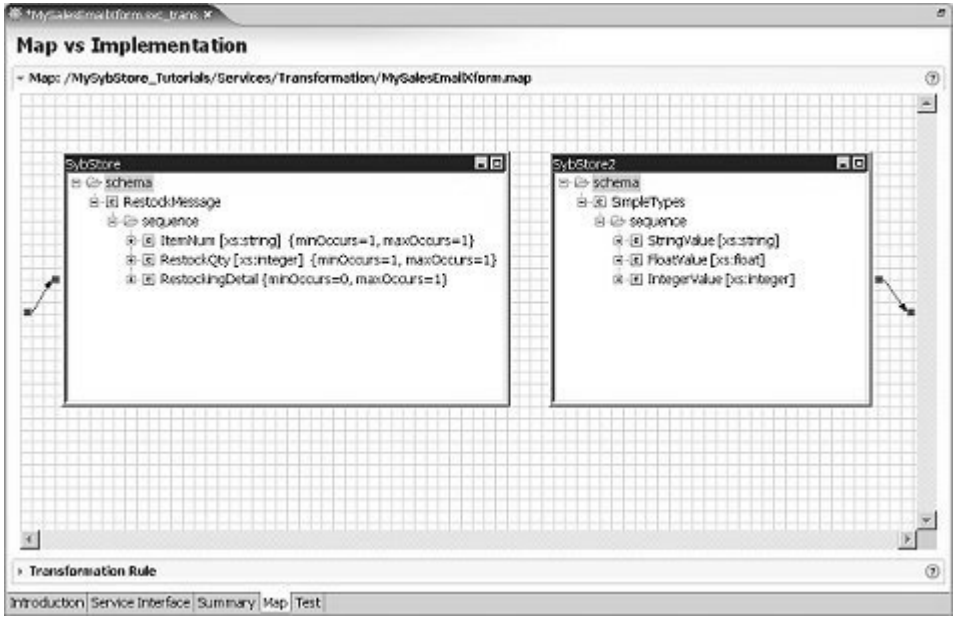
1 Verify that the **Map** tab is selected in the editor. The unmapped source and target models display.



2 Double-click the **SybStore** icon and the **SybStore2** icon to display the schemas in a tree view.

3 Click the maximize button to expand the editor view.

4 Expand the sequences items in both tree views to see all elements in both the source and target schemas.



5 Create the binding between elements in the source schema to elements in target schema:

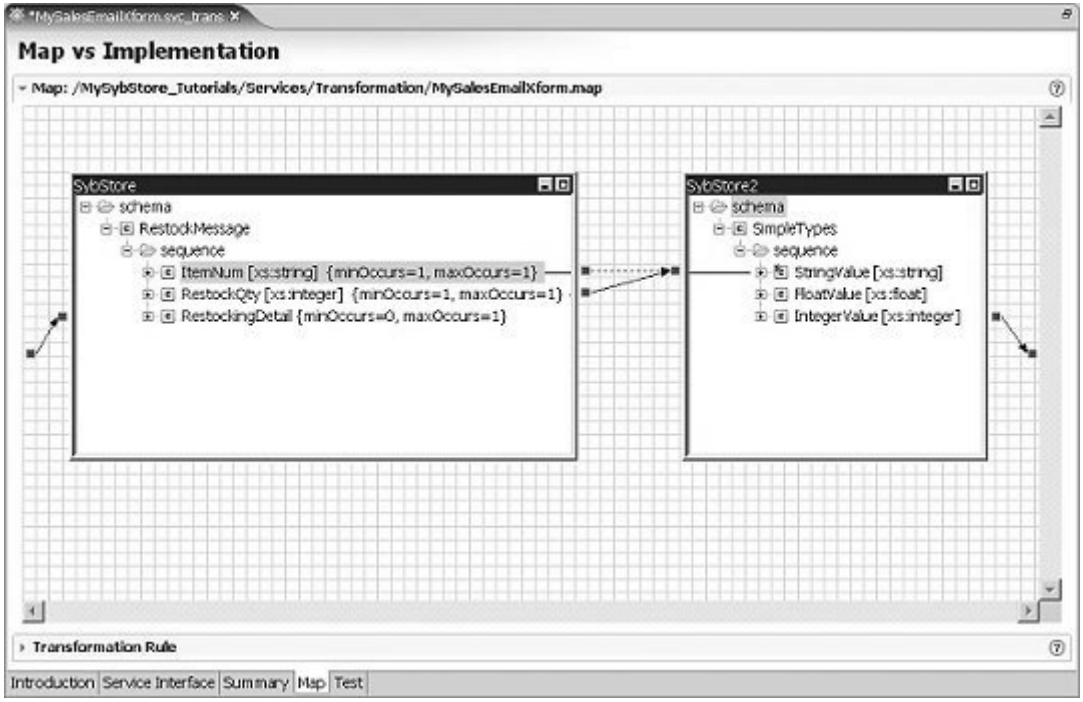
a Drag and drop **RestockQty** from the **Sybstore** tree view onto the **SybStore2** tree value **StringValue**.

a Drag and drop **ItemNum** from the **Sybstore** tree view onto the **SybStore2** tree value **StringValue**.

You see this message:

```
The existing rule on the target node StringValue
will be modified as a result of this action. Do
you wish to continue?
```

b Click **OK**.

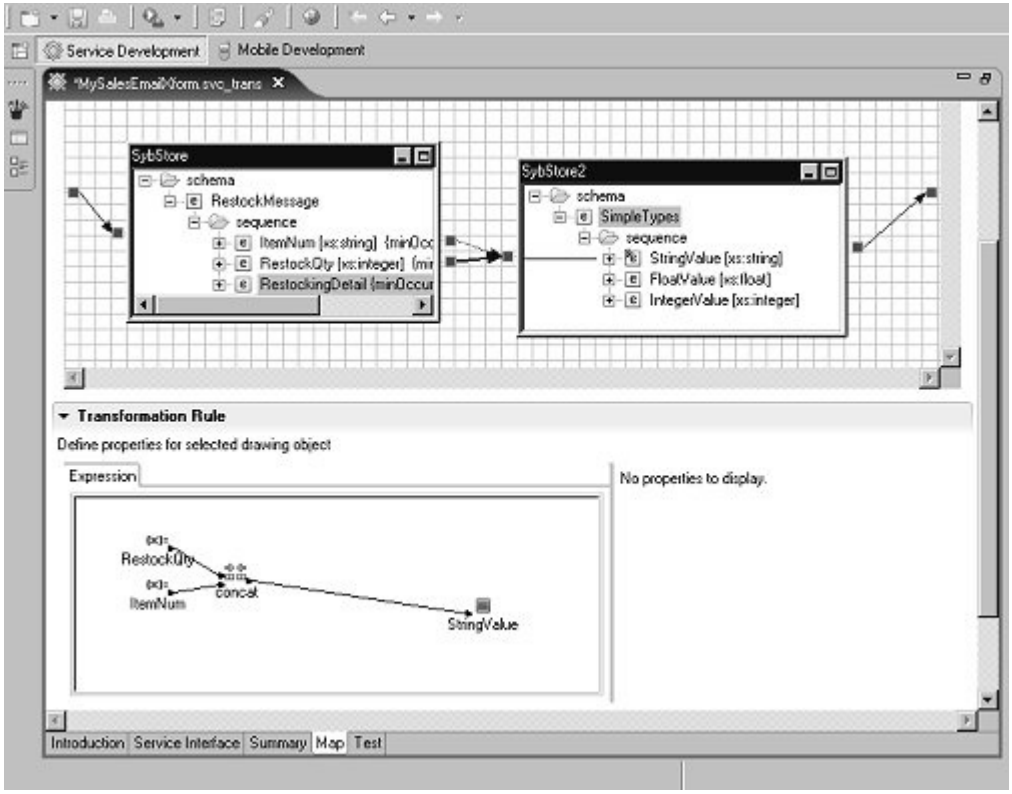


This establishes the base binding between the source and target elements.

6 Click the solid arrow (the bottom line) that connects the source and target schemas. This activates the Expression editor in the Transformation Rule pane below the diagram.

Note There is also a dashed arrow connecting the schemas; ensure you select the solid line.

7 Click the arrow next to the **Transformation Rule** pane title, below the diagram, to expand that section and work in the Expression editor. If you do not see the **Transformation Rule** title, click the arrow beside the **Map** pane to minimize it.

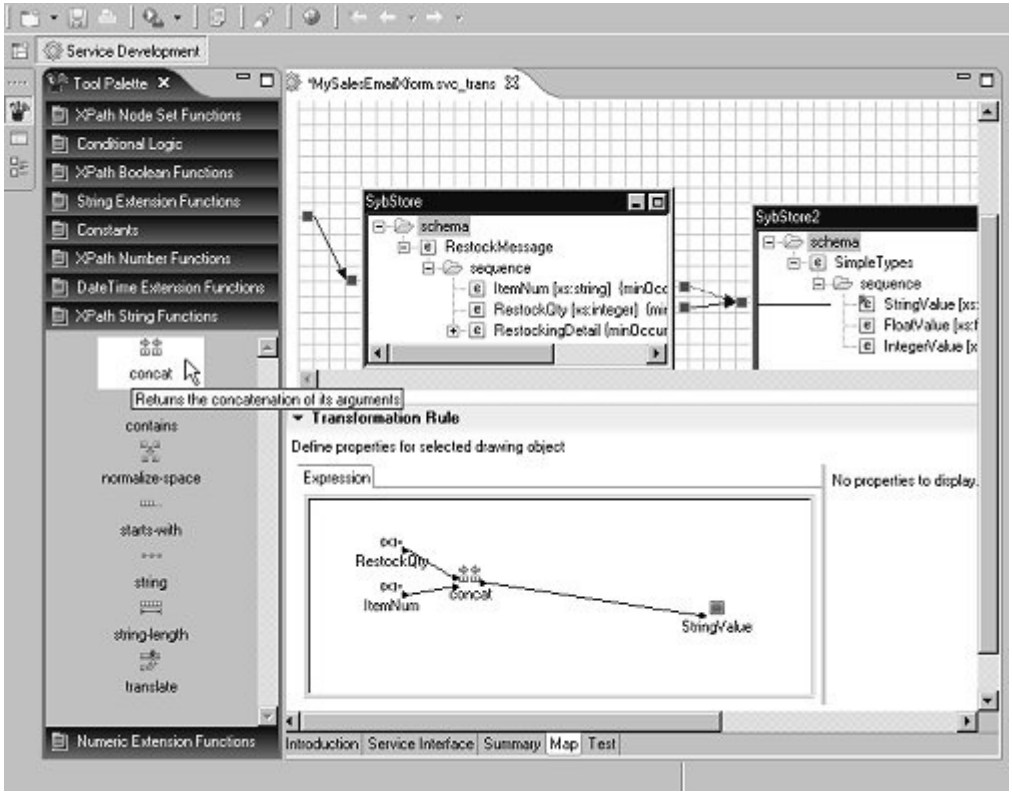


Define additional mapping rules using the Expression editor and the Tool Palette.

8 Select **Window|Show View|Tool Palette** on the WorkSpace main menu.

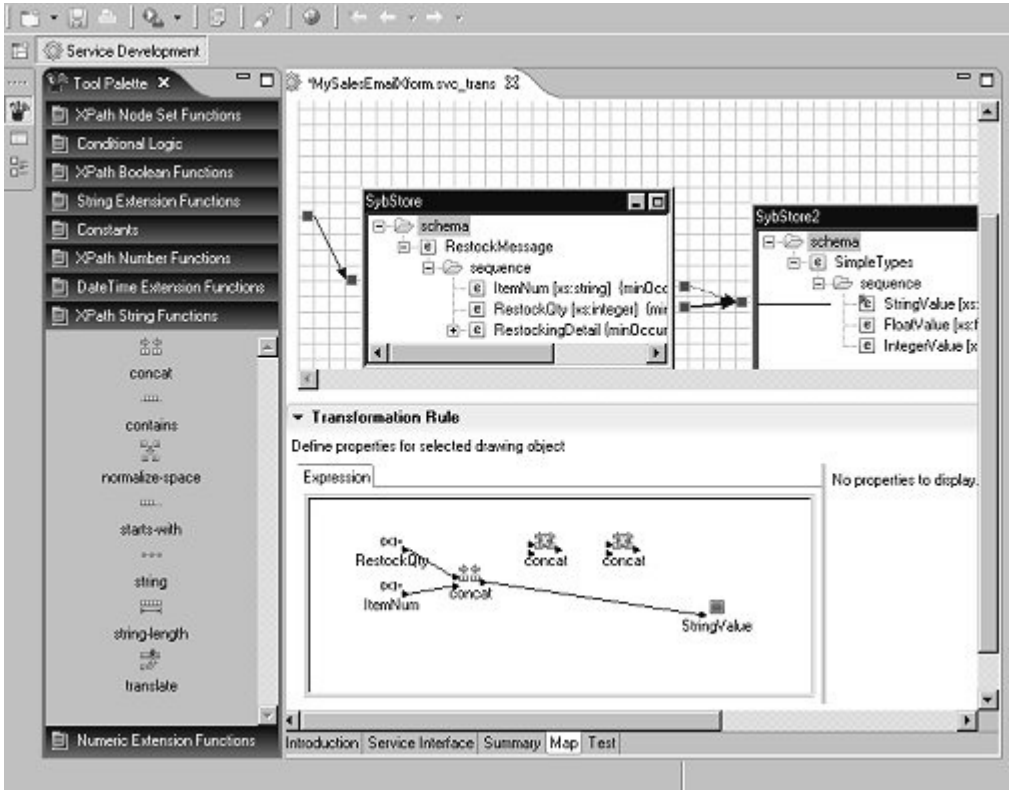
One concat functions already exists in the Expression editor diagram. Add two more.

9 Click in the **Expression** editor pane, select the **Xpath String Functions** category in the **Tool Palette**, select the **concat** function, and drag and drop it onto the **Expression** editor canvas in the **Transformation Rule** pane.



Note If the Tool Palette disappears from view, click the Tool Palette icon in the Fast View to redisplay it.

10 Drag and drop another **concat** function onto the editor canvas.



11 Select the **Constants** category on the **Tool Palette**, select **string constant**, and drag and drop it onto the **Expression** editor canvas.

12 Drag and drop another **string constant** onto the **Expression** editor canvas.

13 To set values for the constant strings:

a Select one of the new **string constant** icons in the **Expression** editor.

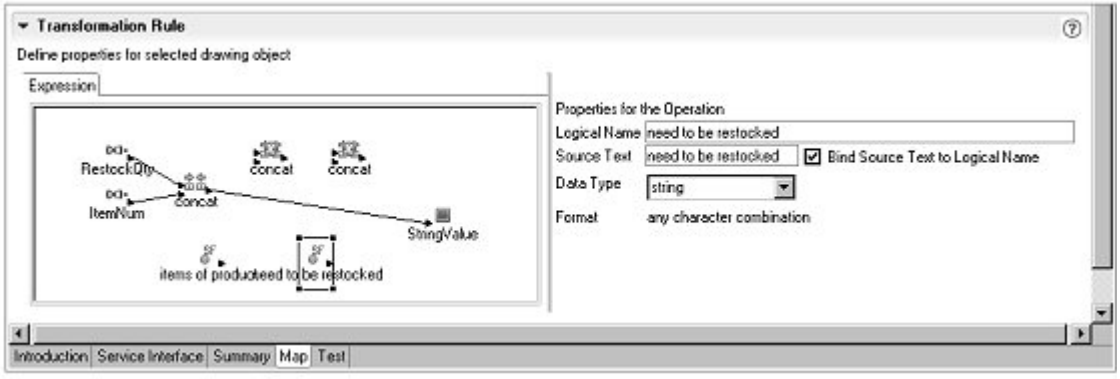
b In the **Properties for the Operation** to the right of the **Expression** editor, change the **Logical Name** to `items of product`.

c Select the **Bind Source Text to Logical Name** option to the right of the **Source Text** field to populate that field with the logical name.

d Select the other new **string constant** icon in the editor.

e Change the **Logical Name** to `need to be restocked`.

f Select the **Bind Source Text to Logical Name** option.



Finish creating the bindings in the Expression editor using the string constants and schema elements.

14 In the **Expression** editor, right-click the line connecting **RestockQty** to **concat** and select **Delete** from the context menu.

15 Right-click the line connecting **ItemNum** to **concat** and select **Delete** from the context menu.

16 Right-click the line connecting **concat** to **StringValue** and select **Delete** from the context menu.

17 In the **Expression** editor, define the transformation mapping from the source schema to the target schema. The small arrows on each object represent input and output links. To connect schemas, click the source arrow on the first object's right side, then drag to and click the target arrow on the other object's left side.

As you work, arrange the icons in the editor to make linking them easier.

a Connect the output link of element **RestockQty** to the upper input link f the first **concat** operation.

b Connect the output link of constant `items of product` to the lower input link of the first **concat** operation.

c Connect the output link of the first **concat** operation to the upper input link of the second **concat** operation.

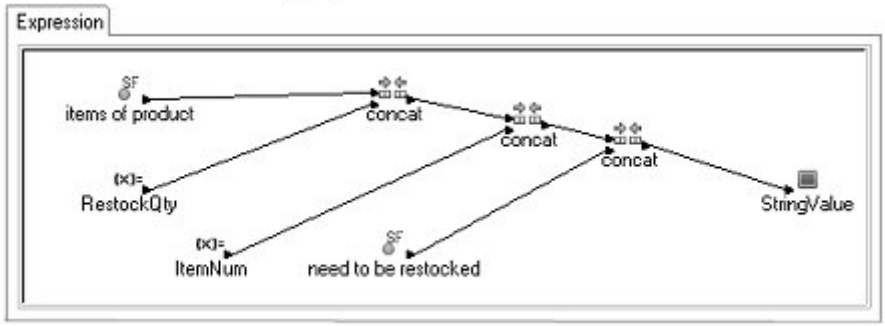
d Connect the output link of the element **ItemNum** to the lower input link of the second **concat** operation.

e Connect the output link of the second **concat** operation to the upper input link of the last **concat** operation.

f Connect the output link of constant `needs to be restocked` to the lower input link of the last **concat** operation.

g Connect the output link of the last **concat** operation to the input link of **StringValue**.

Your expression should have the same connections as the following graphic.



Note If you link incorrectly, right-click the incorrect line and select **Delete** from the context menu, then add the link again.

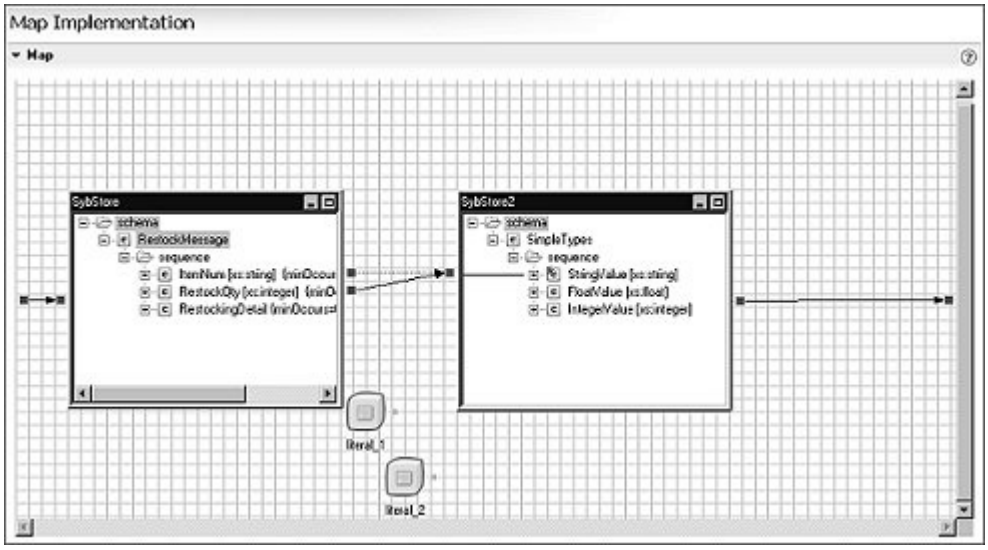
18 The final step is to assign literal values to the remaining target elements so that none of the target elements are unmapped. All fields have a minimum setting of "1" and cannot be "null."

a Expand the **Map** pane (above the Transformation Rule pane) if necessary, then right-click on the canvas and select **Add Literal** from the context menu.

b Right-click on the **Map** canvas again and select **Add Literal** again.

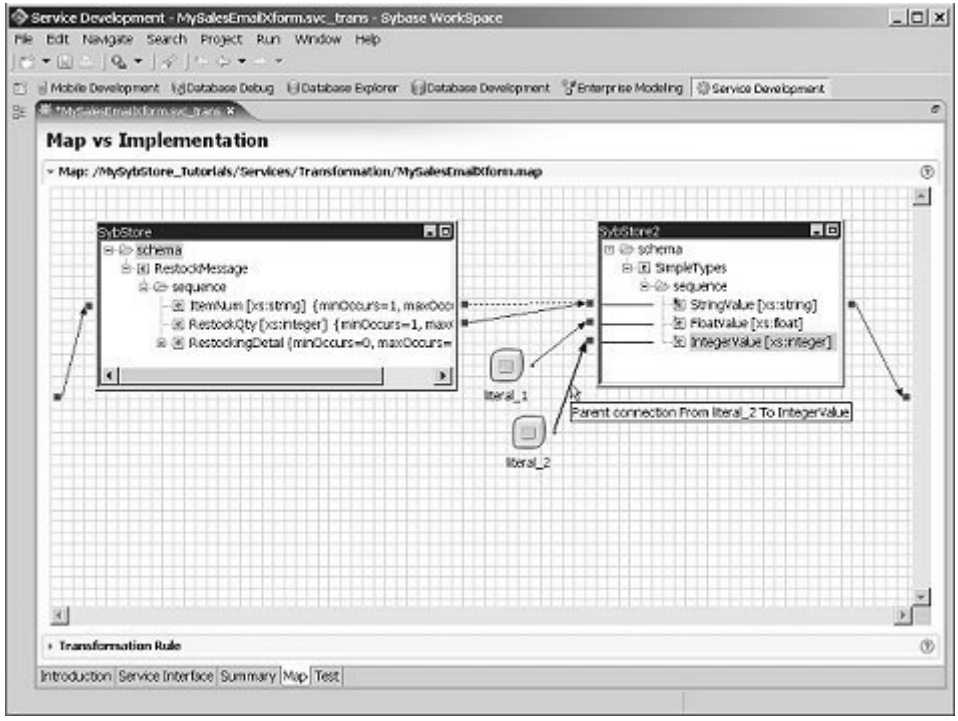
Two literal icons appear in the canvas—`literal_1` and `literal_2`.

c Select each literal icon and move it slightly below and to the left of the **SybStore2** tree view box in the diagram.



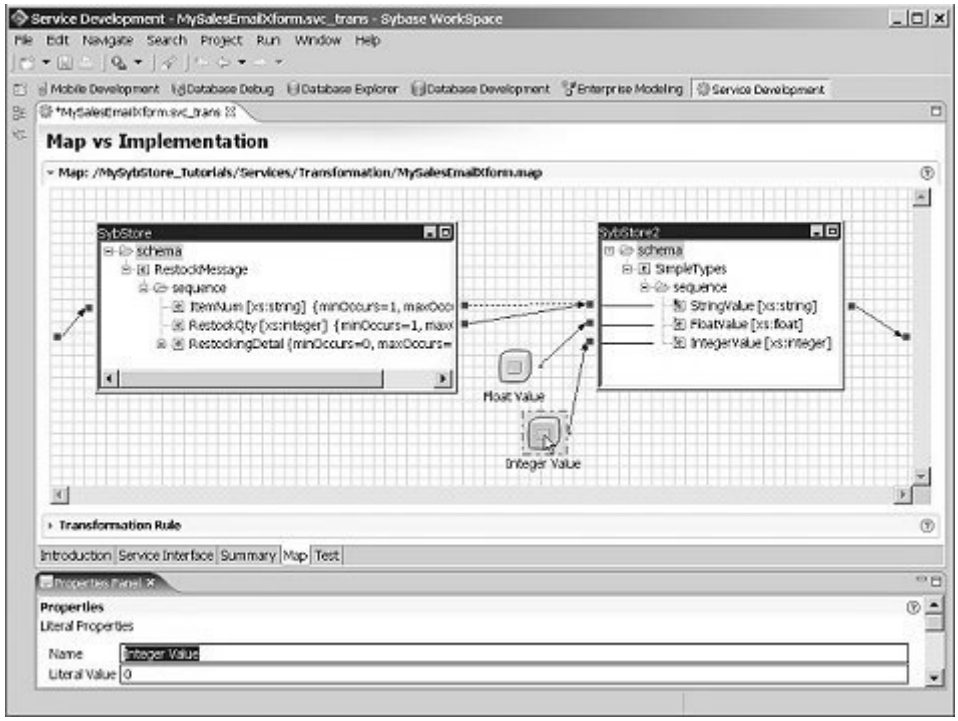
d Click the small connection square for the **literal_1** icon and drag to the **SybStore2** box and click the sequence **FloatValue** to make the connection.

e Click the small connection square for the **literal_2** icon and drag to the **SybStore2** box and click the sequence **IntegerValue** to make that connection.



f Select **literal_1** and, expand the **Properties** pane, change the **Name** to `Float Value` and change **Literal Value** to `0.0`.

g Select **literal_2** and, expand the **Properties** pane, and change the **Name** to `Integer Value` and change its **Literal Value** to `0`.



19 Select **File**|**Save** from the main menu bar.

Leave the new transformation service open in the editor for the next lesson.

You have finished defining the schema mapping for a transformation service. In the next lesson, you test the transformation service using input values from a sample file.

Lesson 3: Testing a transformation service

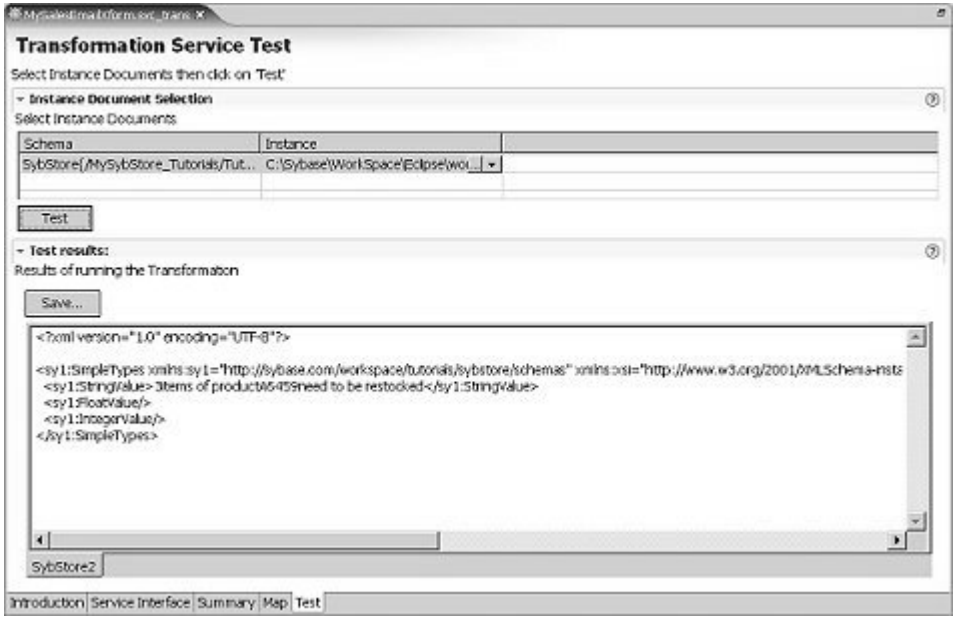
In this lesson, you test a transformation service, using sample data from an XML file.

Before you start this lesson, complete “Lesson 2: Defining mapping for a transformation service” on page 51.

1. Select the **Test** tab at the bottom of the **Transformation Service** editor.
2. In the first row of the **Instance Document Selection** table, click the button with three dots (ellipsis) in the **Instance** column.
3. Select the test data file *%PROJECT_DIR%\MySybStore_Tutorials\Services\Transformation\TestData\DesignTime\RestockMessage.xml* and click **Open**. *%PROJECT_DIR%* is where your personal WorkSpace files are stored.
4. Click **Test** in the **Instance Document Selection** pane.

   If the transformation mapping is correct, these results appear in the **Test Results** pane:

```
<sy1:SimpleTypes xmlns:sy1="http://sybase.com/workspace/
     tutorials/sybstore/schemas">
<sy1:StringValue>3items of productA6459need to be
     restocked</sy1:StringValue>
<sy1:FloatValue>0.0</sy1:FloatValue>
<sy1:IntegerValue>0</sy1:IntegerValue>
</sy1:SimpleTypes>
```



The tested transformation service can be consumed in a business process service.

5 Select **File|Close** from the main menu bar to close the Transformation Service editor.

Creating a message service

A message service enables applications to create, send, receive, and read messages to and from external messaging systems. You can deploy a message service independently, or use it to develop a composite service, such as a business process.

Define message services using simple parameters to describe the message content or by associating the message service with one or more schemas (XSD files) that describe the content of an XML document.

When you develop a message service, you associate it with a messaging endpoint during development, during package definition, or during deployment.

During development, you define a message service as a one-way call to a service implementation via the service interface. The service operations are bound as defined in the services package profile or deployment profile.

When you develop a message service to send data to a transport (outbound service), you define the message service using input style parameters. During deployment, the message service sends the message using the transport to which the service is bound.

This tutorial teaches you how to create a message service using Sybase WorkSpace tools. After you complete this tutorial, you will know how to create and understand the basic components of a message service. You will have a complete service, ready to be deployed and invoked by a business process service.

The message service you create in this tutorial can be invoked by the SybStore application business process service.

This tutorial consists of:

Lesson 1: Creating a message service
Lesson 2: Verifying service parameters
Lesson 3: Specifying e-mail message fields

Lesson 1: Creating a message service

In this lesson, you create a message service for the SybStore application. You configure the new service in subsequent lessons.

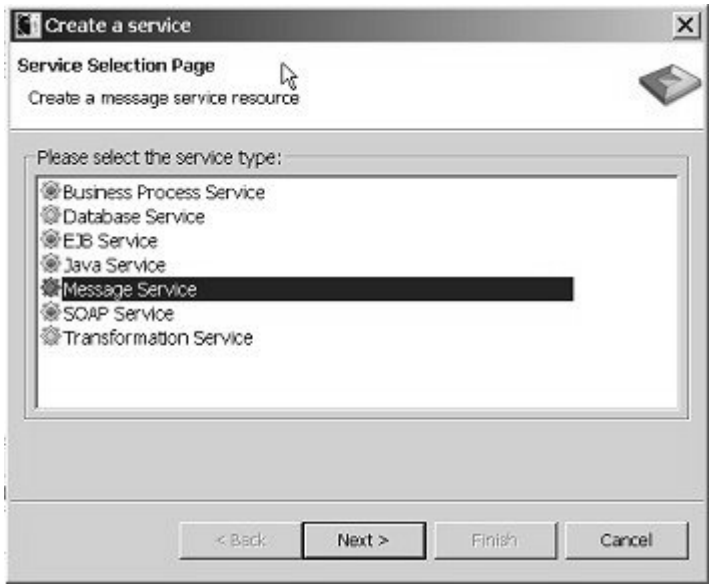
1 Collect and write down the following information from your enterprise's e-mail system administrator. You cannot complete this tutorial without this information.

- SMTP e-mail host
- SMTP e-mail user name
- SMTP e-mail password
- SMTP e-mail port

2 Select **Window|Open Perspective|Service Development** from the main menu bar to open that perspective.

3 Select **File|New|Service** from the main menu bar.

The **Create a Service** wizard appears.



4 Select **Message Service** and click **Next**.

5 Select **MySybStore_Tutorials/Services/Message** to populate the **Enter or Select the Parent Folder** field.

6 Enter `MySalesEmailSend` in the **File Name** field and click **Next**.

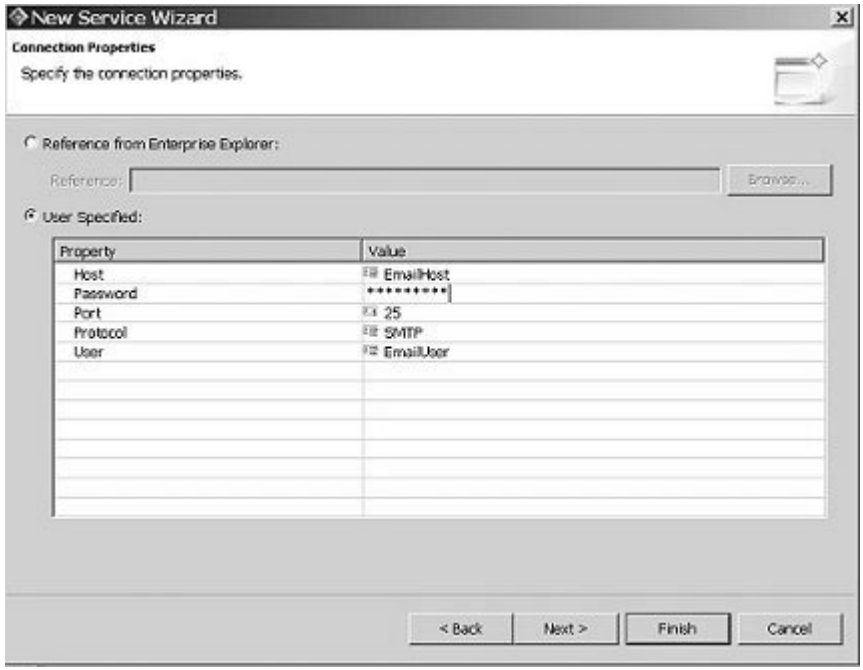
7 On the **Service Summary** page, select **Messaging** as the **Service Information Category** and click **Next**.

8 On the **Service Endpoint Creation** page, select **Yes, Create An Endpoint Now** and click **Next**.

9 Accept the default name "**endpoint**" for the service and click **Next**.

10 On the **Messaging Type** page, select **Email** and click **Next**.

11 On the **Connection Properties** page, select **User Specified** and enter the host, password, port, and user for your e-mail configuration.



12 Click **Finish**. The message service is created and opens in the Message Service Editor.

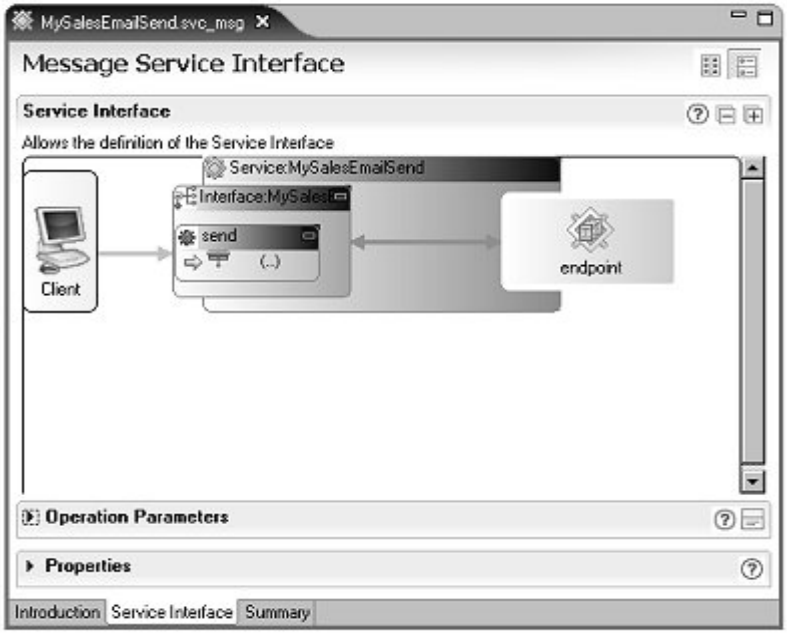
Leave the message service open for the next lesson.

Lesson 2: Verifying service parameters

In this lesson, you verify the service parameters for the message service that you created in the previous lesson.

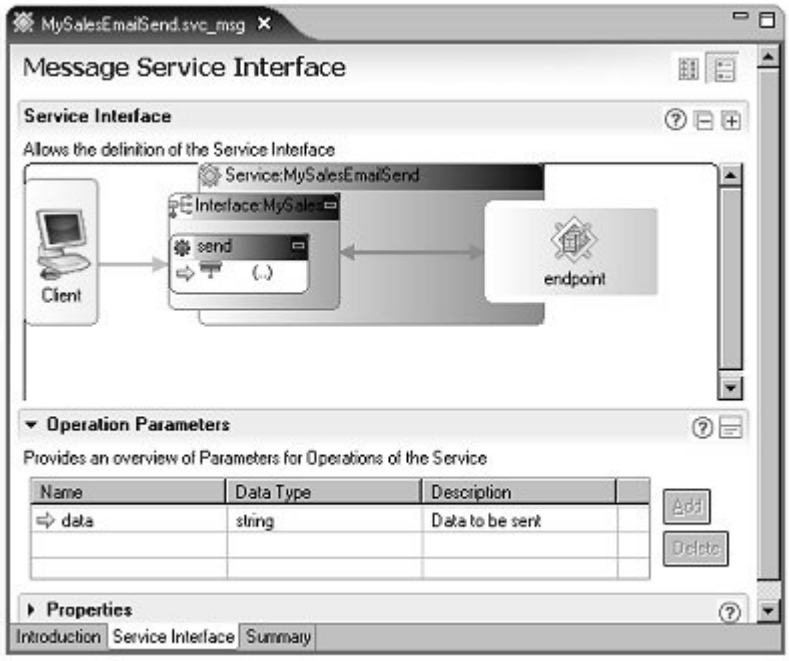
To complete this lesson, you must have completed “Creating a message service” on page 64.

1 Select the **Service Interface** tab in the editor.



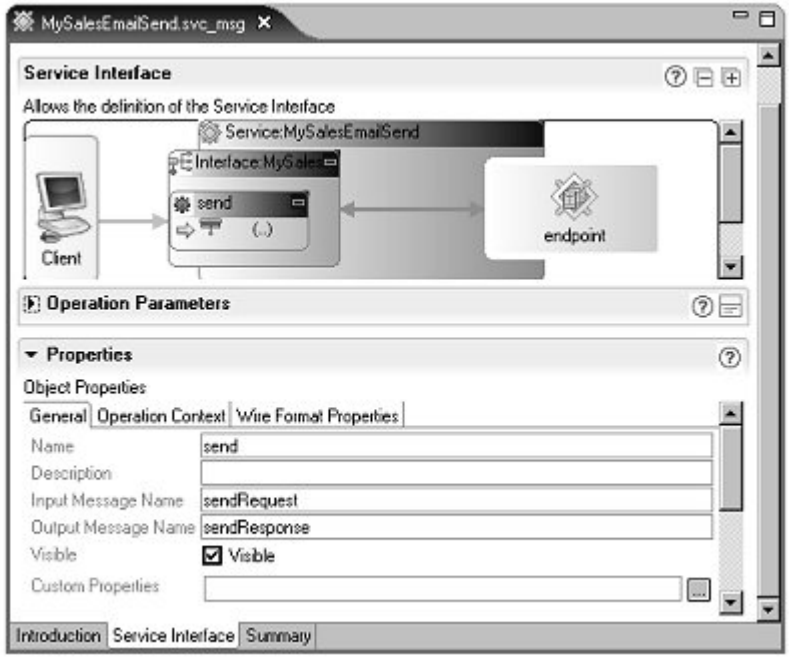
2 Expand the **Operation Parameters** pane by clicking the arrow to the left of the pane's title.

3 Select the **send** operation in the **Service Interface** diagram to see that operation's parameters.



The interface already has the proper parameter. It is a string that contains the text of an e-mail message.

4 Minimize the **Operation Parameters** pane, then expand the **Properties** pane to see the **send** operation's properties.



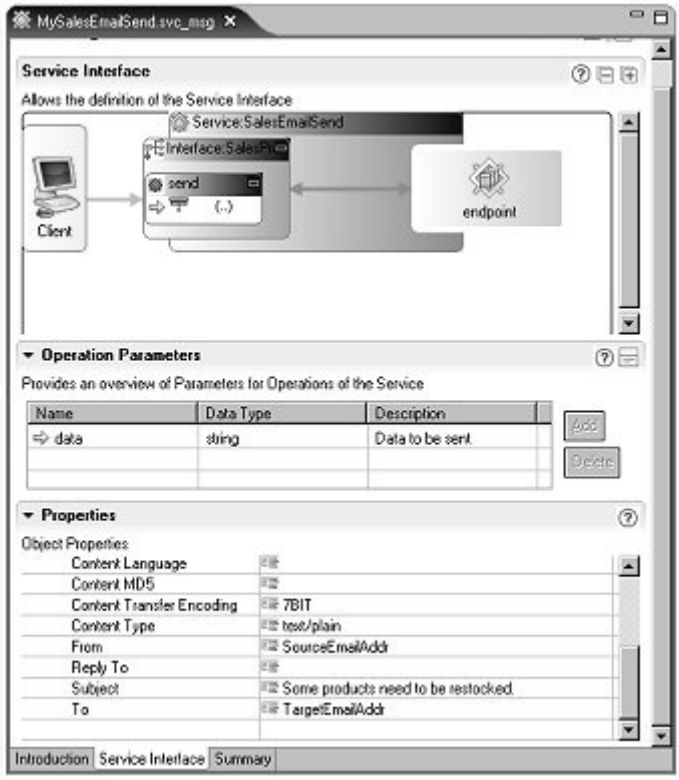
The message service is complete. Leave the message service open for the next lesson.

Lesson 3: Specifying e-mail message fields

In this lesson, you specify the values for the e-mail message fields that you defined in the previous lessons.

1 With **MySalesEmailSend.svc_msg** open in the Message Service Editor, verify that the **Service Interface** tab is selected.

2 In the **Service Interface** diagram, expand the **Properties** pane, then select the **Operation Context** tab to display the e-mail message properties.

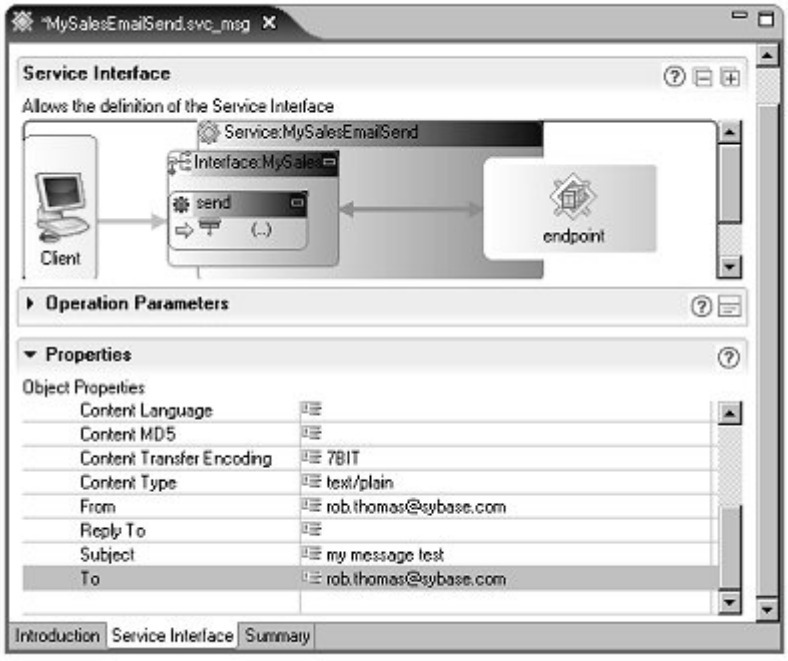


3 Scroll down and enter these values:

- From – the e-mail address from which the message is being sent; for example, `rob.thomas@sybase.com`.
- Reply To – the e-mail address to which a reply is being sent; for example, `sue.smith@sybase.com`
- Subject – enter any subject; for example, `Test e-mail message.`

- To – the e-mail address of the recipient. For a test, enter the same e-mail address entered in the **From** field; for example, `rob.thomas@sybase.com`.

Note The e-mail message values in the From, To, and Reply To values must be in the form of an e-mail address.



4 Select **File**|**Save** from the main menu bar.

5 Select **File**|**Close** from the main menu bar to close the editor.

Using a generic JMS provider for messaging

Sybase WorkSpace and Unwired Orchestrator allow you to use JMS resources provided by a generic messaging provider (other than a default WorkSpace messaging provider) for messaging transports.

This enables you to use JMS resources provided by a generic messaging provider. You must supply the client JAR files, and configure a connection profile to the provider.

Note A generic JMS provider must adhere to the JMS specification for interfacing with messaging transports.

Details about the JMS specification for interfacing with messaging transports are available from the Java Developers Network at http://java.sun.com/products/jms/.

This tutorial shows you how to properly configure a connection profile to a generic JMS messaging transport and to create a messaging service. The tutorial uses the BEA WebLogic Server version 9.0 as an example of a generic JMS provider, but you can use any third-party provider.

This tutorial contains:

Lesson 1: Specifying the location of generic JMS client JAR files
Lesson 2: Configuring generic JMS transport connection profile
Lesson 3: Creating a generic JMS transport messaging service

Lesson 1: Specifying the location of generic JMS client JAR files

In this lesson, you specify the location of the client JAR files for the generic JMS provider.

Note You must have a third-party generic JMS provider installed, such as BEA WebLogic Server, to complete this lesson.

---

1 In WorkSpace, select **Window|Preferences** from the main menu toolbar.

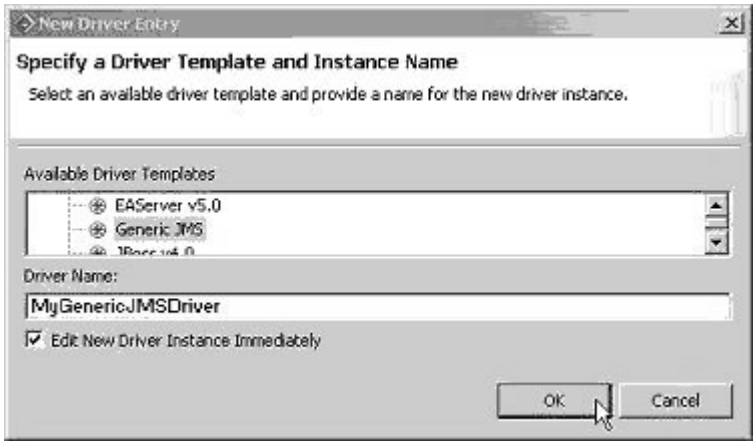
2 When the **Preferences** dialog box opens, in the left pane expand **Sybase, Inc.**, then select **Driver Definitions**.

3 In the **Preferences** pane under **Available Driver Definitions**, select the **JMS** category and click **Add**.

4 When the **New Driver Entry** dialog box opens, select **Generic JMS** in the **Available Driver Templates** list.

5 Enter `MyGenericJMSDriver` in the **Driver Name** field, select **Edit New Driver Instance Immediately**, and click **OK**.

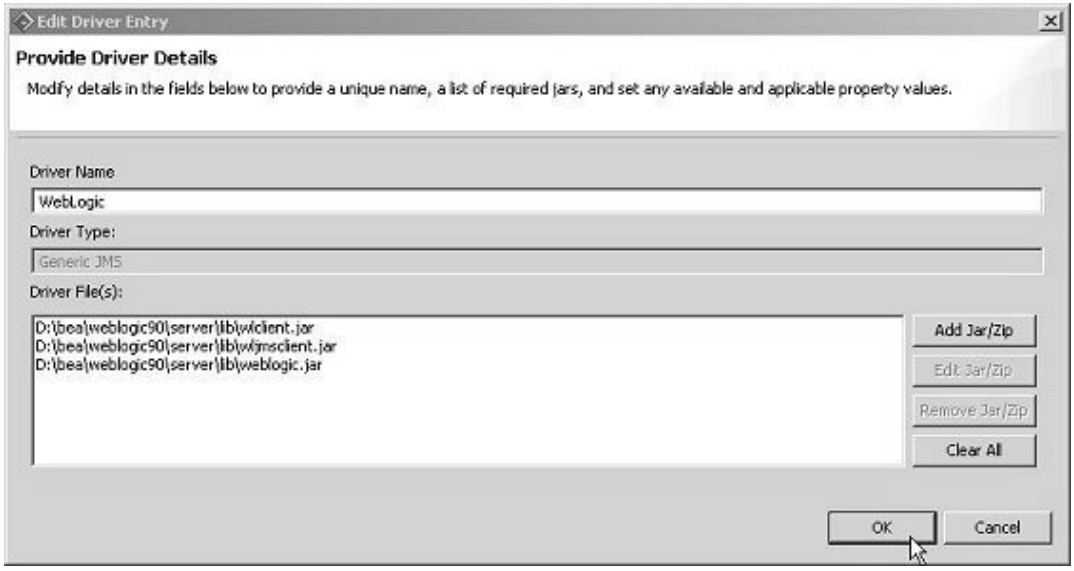
Note `MyGenericJMSDriver` is an example driver name. You can enter any name you choose.



6 When the **Edit Driver Entry** window displays, click **Add Jar/Zip**.

7 In the file selection window, go to the directory where the generic JMS provider's client JAR files reside and select the JAR files until you have a complete list, then click **OK**.

The example below shows the names and locations of BEA WebLogic client JAR files. Use the names and locations of the client JAR files for the generic JMS provider you are using.



8 Click **OK** to close the **Preferences** dialog box.

You have specified the locations of the client JAR files for a generic JMS provider. Next, you can create a connection profile for the provider.

Lesson 2: Configuring generic JMS transport connection profile

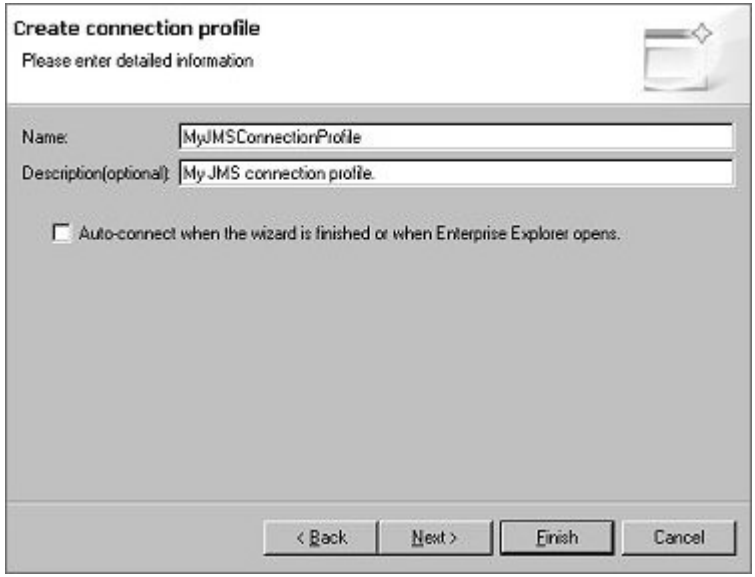
In this lesson, you configure a connection profile to a generic JMS messaging transport provider. Before you begin, you must complete “Lesson 1: Specifying the location of generic JMS client JAR files” on page 73.

1 Select **Window|Show View|Enterprise Explorer**.

2 In the **Enterprise Explorer**, right-click the **Message Transports** folder and select **New**.

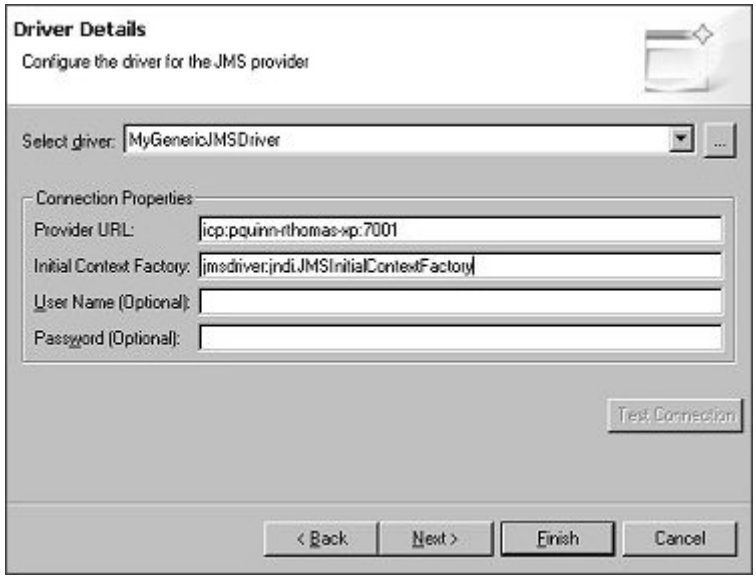
3 When the **New Connection Profile** dialog box opens, select **JMS** and click **Next**.

4 On the **New JMS Connection Profile** page, enter a **Name** for this connection profile; for example `MyJMSConnectionProfile`. You can also type an optional description for the JMS provider.

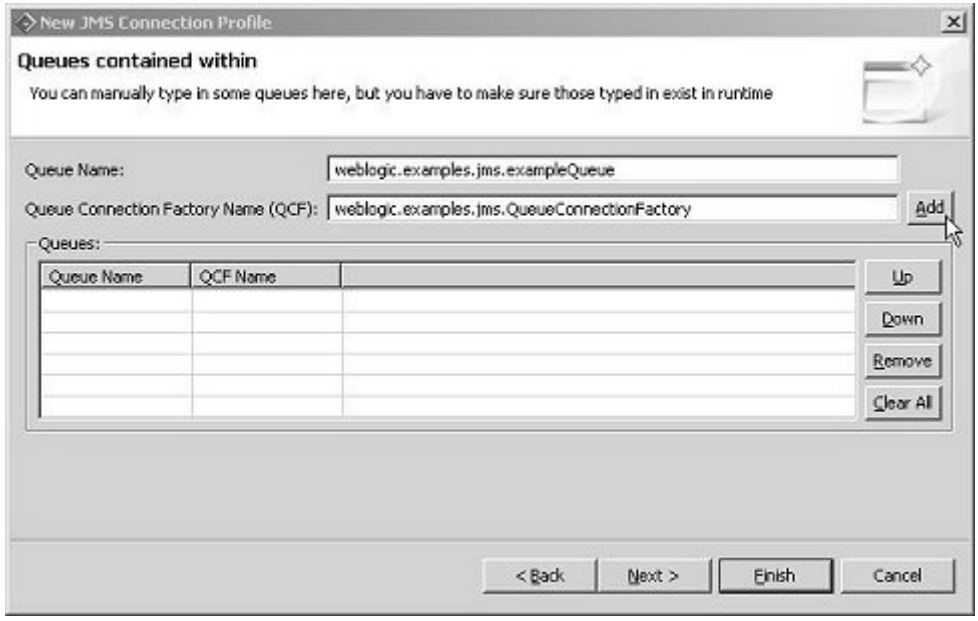


5 Click **Next**.

6 On the **Driver Details** page, choose the name of the driver that you specified in "Lesson 1: Specifying the location of generic JMS client JAR files" on page 73 from the **Select driver** list, fill in the remaining fields as appropriate, then click **Next**.



7 When the **Queues Contained Within** page opens, specify the connection parameters for a queue hosted by the provider. The screen below shows the connection parameters for a BEA WebLogic server hosted queue; choose the connection parameters for a queue hosted by the particular generic JMS provider you are using:



8 Click **Add**. Continue to add any other queues as necessary, and click **Finish**.

You have specified the connection parameters for a queue hosted by a generic JMS transport provider. Now you can create a message service from this queue definition.

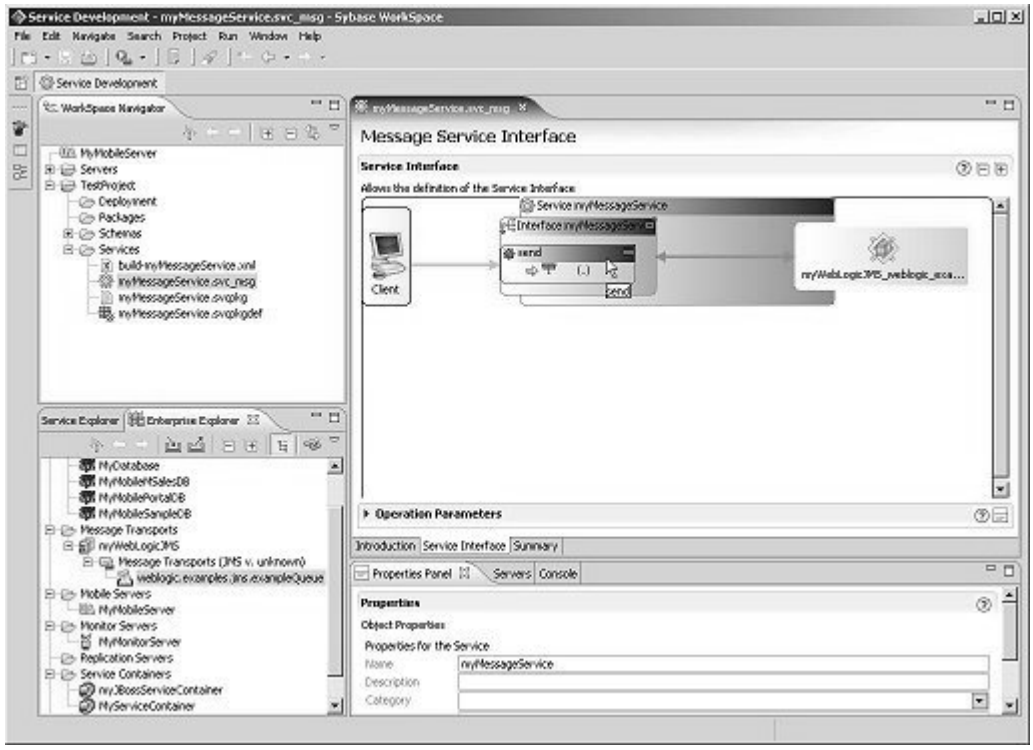
Lesson 3: Creating a generic JMS transport messaging service

In this lesson, you create a messaging service that uses the generic JMS transport as its endpoint that you configured in the previous lesson.

1 In the **Enterprise Explorer** view, right-click the generic JMS queue you configured in lesson 2 of this tutorial, and select **Create Message Service**.

2 In the **New Service Wizard**, select the location in which to save the message service, which displays in the parent folder field, enter a message service name, then click **Finish**.

3 The new service is created and opens in the Message Service Editor.



4 Select **File|Save** from the WorkSpace main menu, then select **File|Close**.

You now know how to specify the location of the client driver JAR files for a generic JMS provider, configure a connection profile to a generic JMS queue, and create a message service from the queue definition.

Packaging, deploying, and testing a service

Sybase WorkSpace supports many packaging strategies and options. This tutorial guides you through a simple example of service packaging, deployment, and testing.

In general, a service package profile allows you to define the services to be included in a package, and the configuration for endpoint and runtime specifications. The package profile is a reference to the included services, but does not actually include the services.

After you complete this tutorial, you will know how to package, test, and deploy a service, and you should understand the basic concepts of service packages. You will have a complete service package, ready to be deployed.

This tutorial consists of:

Lesson 1: Creating a package profile
Lesson 1: Changing the logging level and packaging the service
Lesson 3: Deploying a service
Lesson 4: Testing a service

Lesson 1: Creating a package profile

During the packaging phase, you develop a package profile, which is used to create the physical package that is deployed in a runtime environment.

1 Select **Window|Open Perspective|Service Development** from the main menu bar.

2 In the **WorkSpace Navigator**, expand the **MySybStore_Tutorial/Services/Message** folder.

3 Right-click **MySalesEmailSend.svc_msg** and select **Create Sybase Services Package Profile** from the context menu.

4 The new package profile is created and opens in the Sybase Services Package Profile Editor.

Lesson 1: Changing the logging level and packaging the service

In this lesson, set the log level to simplify debugging the service after it is deployed, then you package the service.

Service logging messages are sent to the runtime container log file. For Unwired Orchestrator and EAServer, this is the *Jaguar.log* file in the *%WORKSPACE_DIR%\DevRuntimes\EAServer\bin* directory.

1 Select **Window|Open Perspective|Service Development** from the main menu bar.

2 In the **WorkSpace Navigator**, expand the **MySybStore_Tutorial/Services/Message** folder.

3 Double-click **MySalesEmailSend.svcpkgdef** to open the file in the package profile editor.

4 To change the logging level, select the **Runtime Container Configuration** tab at the bottom of the editor.

5 Select **FINER** in the **Log Level** list.

6 Click **File|Save** on the WorkSpace main menu to save the change.

7 In the **WorkSpace Navigator**, expand the **MySybStore_Tutorial/Services/Message,** right-click **MySalesEmailSend.svcpkgdef**, and select **Build Package** from the context menu.

8 You see the progress of the build in the Console view. When a message states that the package was built successfully, click **OK**.

9 Click **File|Save** on the WorkSpace main menu to save the package profile.

10 Select **File|Close All** on the WorkSpace main menu to close the editor and any other open views.

Lesson 3: Deploying a service

This lesson shows you how to deploy the service package that you created in the previous lessons.

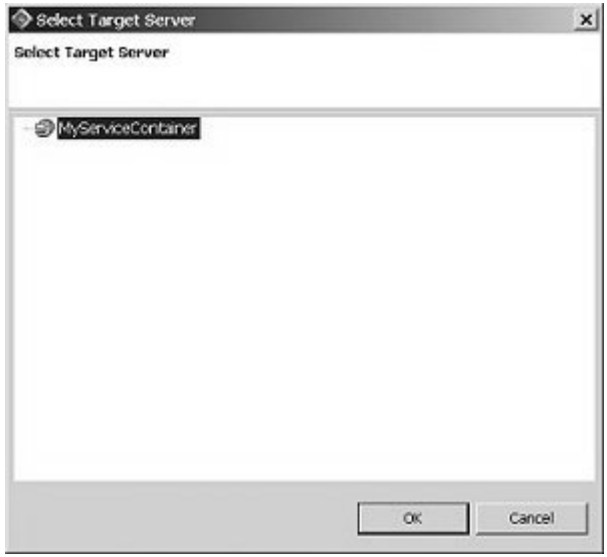
Note Before you begin, Unwired Orchestrator must be running and you must have a connection established to the server from the MyServiceContainer connection profile in WorkSpace. See "Starting and connecting to the Unwired Orchestrator server" on page 19 for instructions.

---

1 In the **WorkSpace Navigator**, expand **MySybStore_Tutorials/Services/Message**.

2 Right-click **MySalesEmailSend.svcpkgdef** and select **Deploy Package** from the context menu.

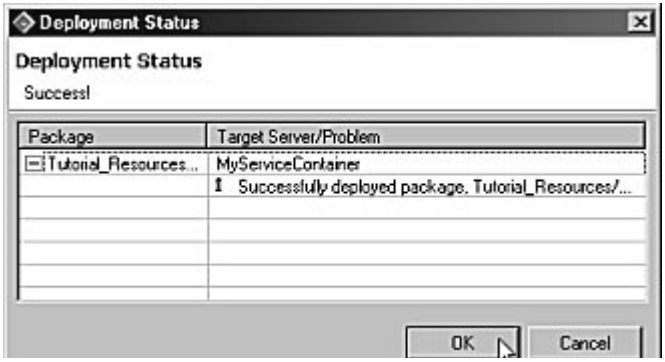
3 When the **Select Target Server** dialog box opens, select **MyServiceContainer** as the target server (runtime container) to which you want to deploy the service and click **OK**.



The services package profile definition is used to build the package. The progress is shown in a Console window. If you see a message asking if you want to overwrite an existing package, answer yes.

When the package builds successfully, it is deployed to the MyServiceContainer server.

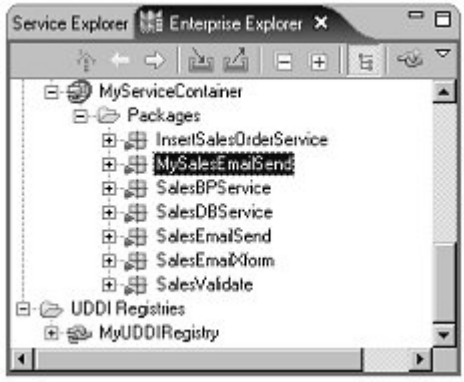
4 When the **Deployment Status** message states that the deployment was successful, click **OK**.



5 Click **"X"** on the **Console** title tab to close that view.

After you successfully deploy the services package to a runtime container, you can view the package in the **Enterprise Explorer**.

6 In the **Enterprise Explorer**, expand **Service Containers/MyServiceContainer/Packages** and verify that the **MySalesEmailSend** package displays.



7 To review the EAServer log, connect to EAServer (which starts when you start Unwired Orchestrator) using the **MyAppServer** connection profile.

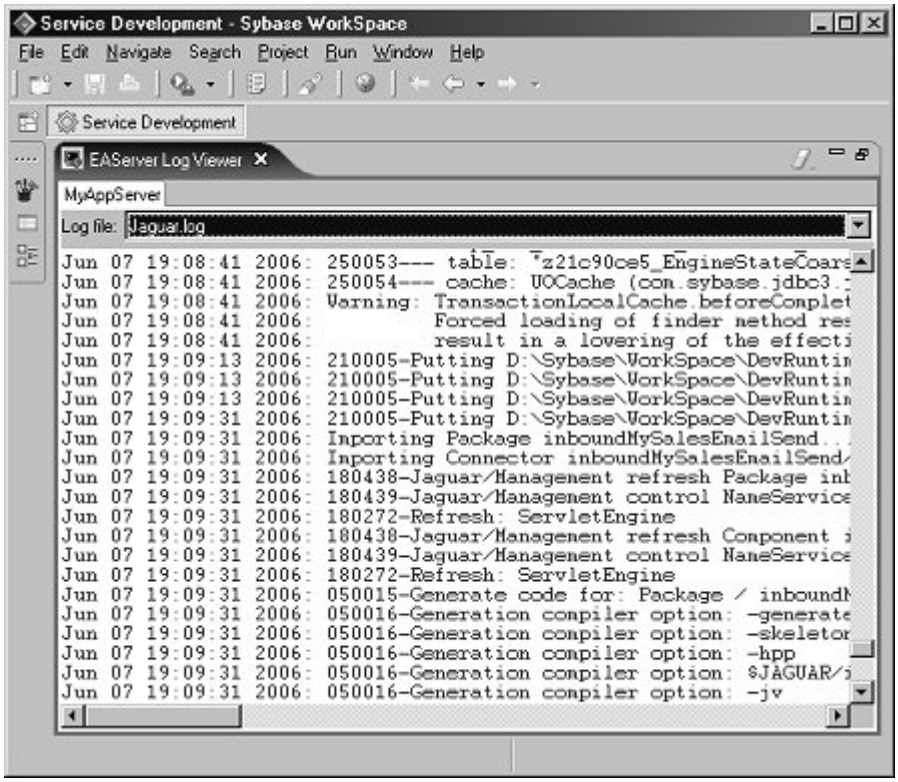
8 In the **Enterprise Explorer**, expand the **Application Servers** folder, right-click on **MyAppServer**, and select **Connect**.

9 To see the log file, select **Window|Show View|Other** from the main menu bar.

10 In the **Show View** dialog box, expand the **Sybase** folder and select **EAServer Log Viewer** and click **OK**.

11 When the **EAServer Log Viewer** opens, maximize the window and select **Jaguar.log** from the **Log File** drop-down list.

12 When the log displays, review the entries for the package deployment.



13 Click the "**X**" on the **EAServer Log Viewer** title tab to close the view.

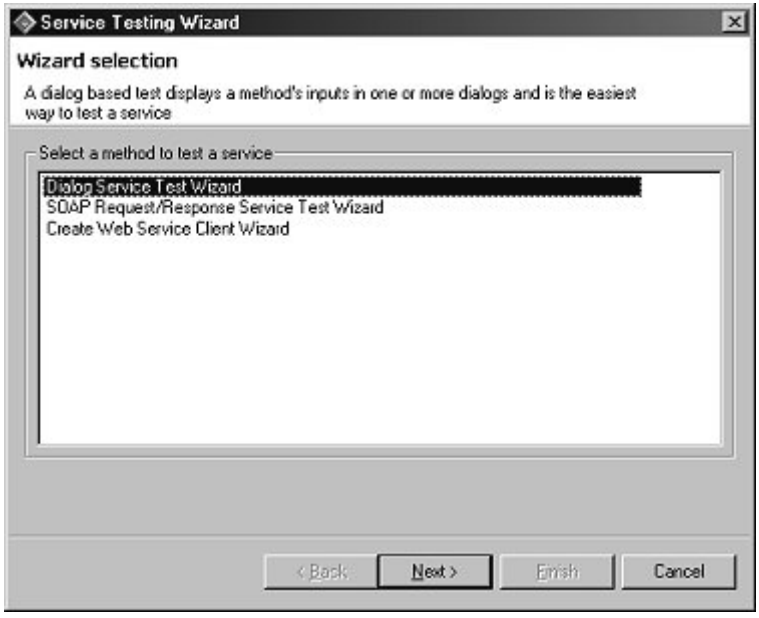
Lesson 4: Testing a service

This lesson teaches you how to test the deployed service. There must be a connection established to Unwired Orchestrator from the MyServiceContainer connection profile in WorkSpace.

1 In the **Enterprise Explorer**, expand **Service Containers/MyServiceContainer/Packages/MySalesEmailSend/Services**.

2 Right-click **MySalesEmailSend** and select **Test Service** from the context menu.

The Service Testing Wizard opens.

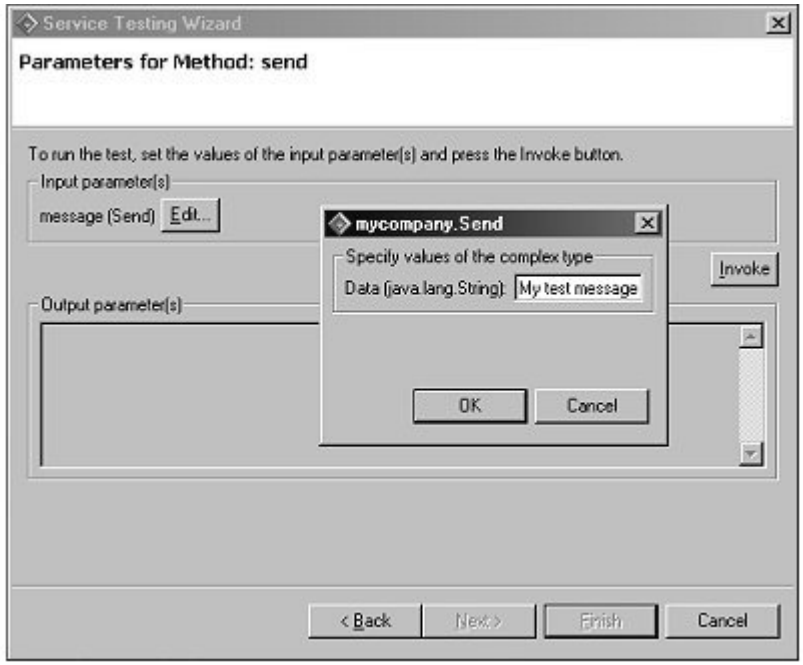


3 Select **Dialog Service Test Wizard** and click **Next**.

4 On the **Options** page, click **Next** to accept the default selections.

5 On the **Select A Method To Test** page, select the `void send(Send message)` method and click **Next**.

6 On the **Parameters for Method** page, click **Edit**, enter `My test message` in the **Data** field of the pop-up dialog box, and click **OK**.



7 Click **Invoke** to start the service, sending the string you entered.

Because the service does not have an output parameter, no value appears in the **Output parameter(s)** pane. However, a message does appear at the bottom of the window indicating the method was successfully invoked.

Note If you entered valid e-mail parameters, you should receive an e-mail message with the subject line entered in "Lesson 3: Specifying e-mail message fields" on page 70 and content of `My test message`.

---

8 Click **Finish** to close the **Service Tester**.

9 To check for errors in the log file, select **Window|Show View|Other** from the main menu bar.

10 In the **Show View** dialog box, expand the **Sybase** folder and select **EAServer Log Viewer** and click **OK**.

11 When the **EAServer Log Viewer** opens, maximize the window and select *Jaguar.log* from the **Log File** drop-down list.

12 Select **Window|Close All Perspectives**.

Congratulations! You have completed the Service Development tutorials.

CHAPTER 3

Process Orchestration Tutorials

The Process Orchestration tutorials teach you how to create and debug a simple business process service.

Topic	Page
Overview	89
Creating a simple business process service	90
Packaging, deploying, and testing a business process service	143
Debugging a business process service	151
Creating a business process service correlation set	166

Overview

Sybase WorkSpace supports the development of composite business process services and allows you to extend its reach in the enterprise by integrating alert messaging, business activity monitoring, custom wire formats, and database event management.

The tutorials in this chapter teach you how to develop, package, deploy, test, and debug a simple business process service. You also learn how to create a correlation set for a business process service. Correlation sets allow you to define properties that allow data defined in the initial message to match data defined in a subsequent message.

For detail information, see the online help topics *Sybase WorkSpace Development|Process Orchestration* and *Defining Business Process Correlation Sets* under *Sybase WorkSpace Development|Develop|Developing a Business Process Service|Designing a Business Process*.

Prerequisites

Before you begin the Process Orchestration tutorials, complete the procedures in Chapter 1, “Introduction, Installation, and Setup.”

Creating a simple business process service

This tutorial shows you how to create a simple business process service and add some activities to it: a service invocation, a simple rule for branching, and error handling.

Note This tutorial uses a "bottom-up" approach. To learn how to create a business process service using a "top-down" approach, see "Generating a Business Process service" on page 61 in the *Sybase WorkSpace SybStore Tutorials: Enterprise Modeling* guide.

This tutorial consists of:

Lesson 1: Creating a business process service
Lesson 2: Adding a service invocation to a business process service
Lesson 3: Adding a rule to a business process service
Lesson 4: Defining error handling for a business process service
Lesson 5: Setting message context properties dynamically

Lesson 1: Creating a business process service

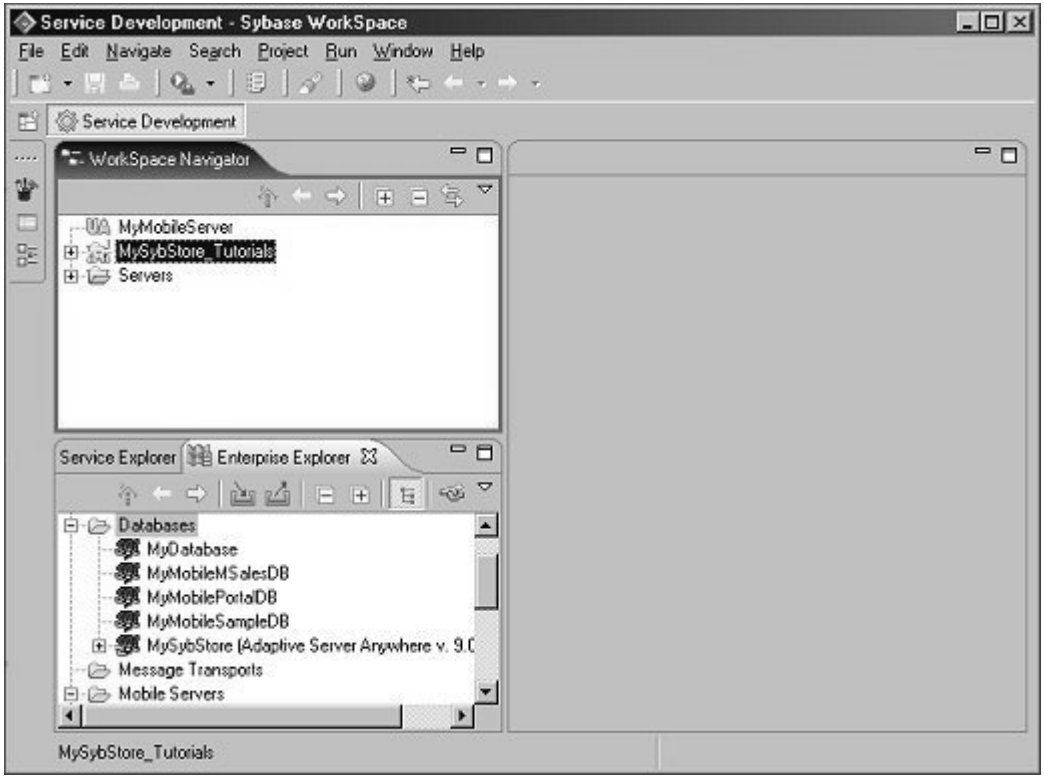
In this lesson, you create a simple business process service called MySalesBPService. The service has one operation—ManageInventory.

The ManageInventory operation takes an input parameter called inputSalesItem, and returns an output parameter called outputResponse. The *inputSalesItem* variable contains information about a sale—item identifier, quantity sold, price, and so on. The *outputResponse* variable contains status information about the execution of the service—success or failure, and if there is a failure.

1 Select **Start|Programs|Sybase|Sybase WorkSpace|Sybase WorkSpace 1.5**.

2 Select **Window|Open Perspective|Other**, select **Service Development**, and click **OK**.

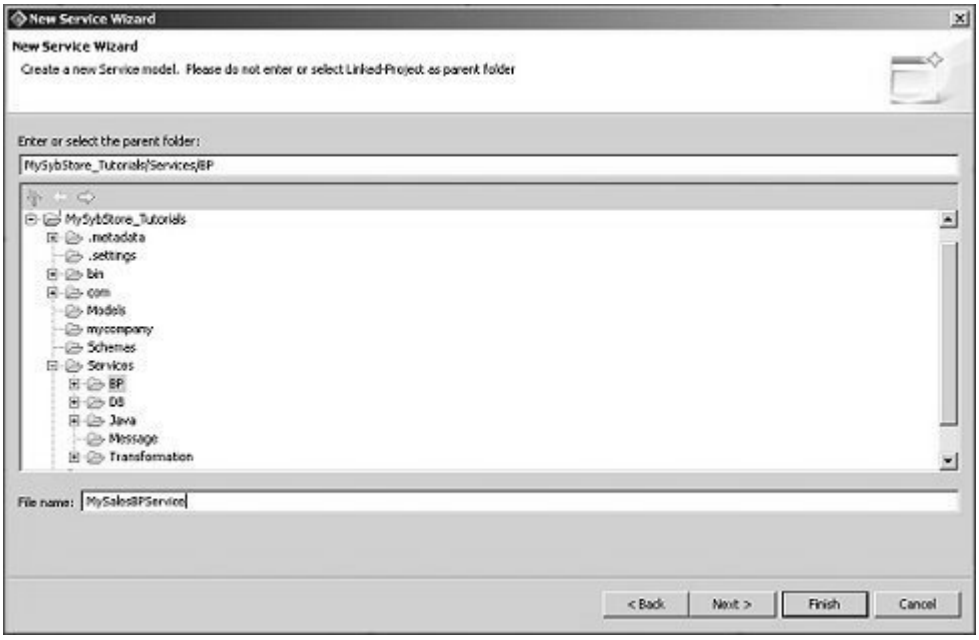
The Service Development perspective opens the WorkSpace Navigator, the Service Explorer, and the Enterprise Explorer views.



3 In the **WorkSpace Navigator**, right-click the **MySybStore_Tutorials/Services/BP** folder and select **New|Service** from the context menu.

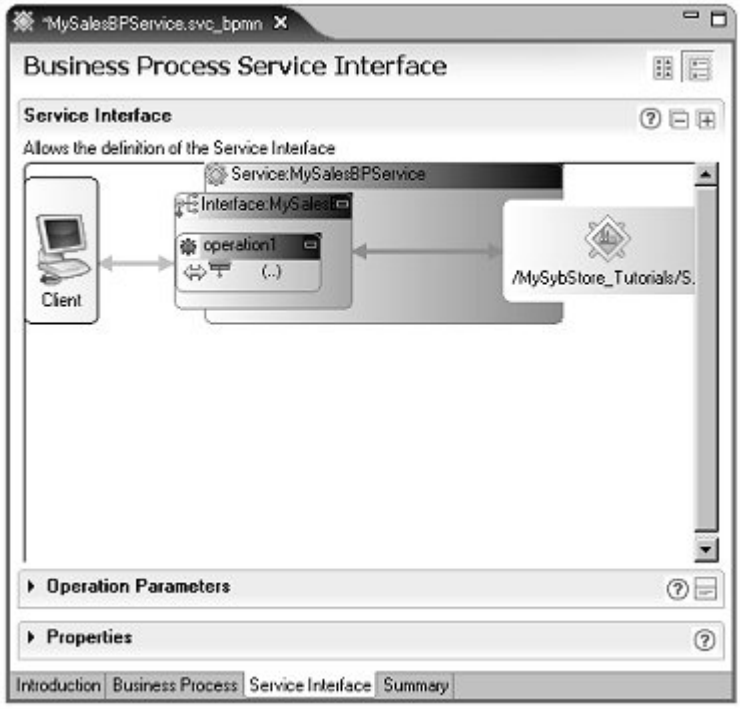
4 In the **Creat a Service** dialog box, select **Business Process Service** and click **Next**.

5 Verify that **MySybStore_Tutorials/Services/BP** is selected as the parent folder, then enter `MySalesBPService` for the **File Name** and click **Finish**.



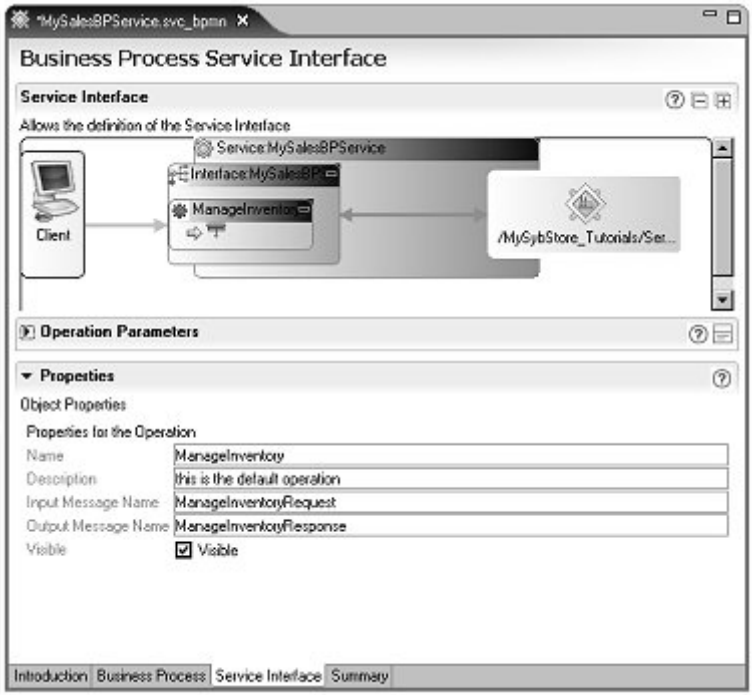
The new service is created and opens in the Business Process Service Editor.

6 In the editor, select the **Service Interface** tab.

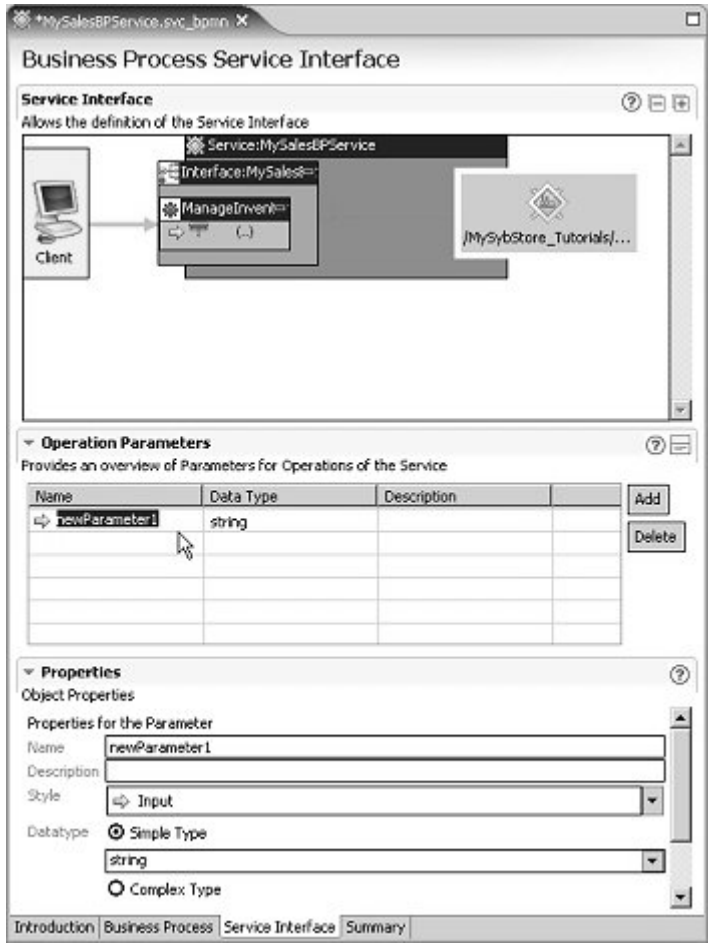


7 In the **Service Interface** diagram, expand the **Properties** pane (at the bottom of the editor) and select the **operation1** box in the editor diagram.

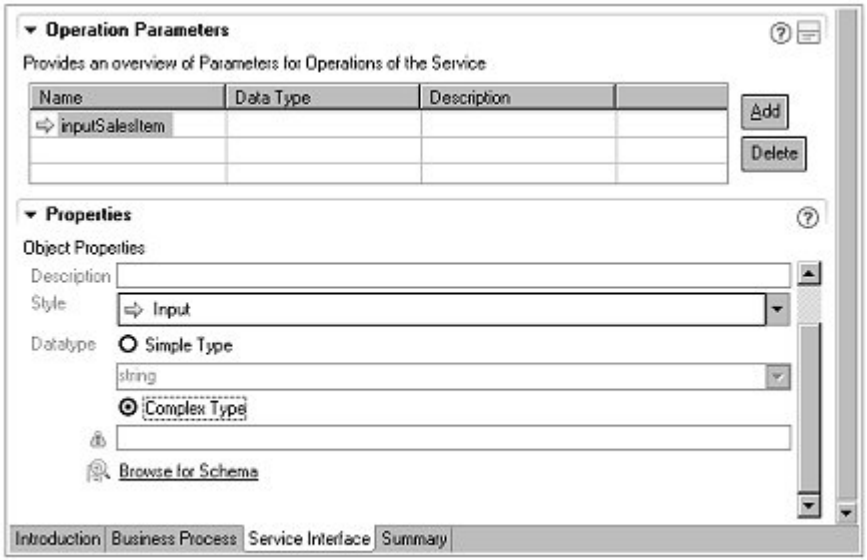
8 Enter `ManageInventory` in the **Name** field. The Input Message Name and Output Message Name fields update automatically to reflect the change.



9 Expand the **Operation Parameters** section and click **Add** to add a new input parameter.

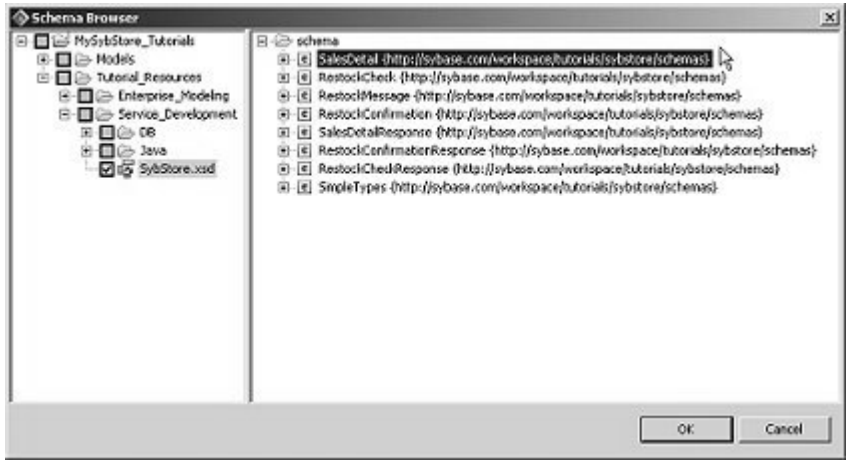


10 Select the new input parameter in the **Operation Parameters** pane. In the **Properties** pane, enter `inputSalesItem` in the **Name** column, and select **Complex Type** as the **Datatype**.

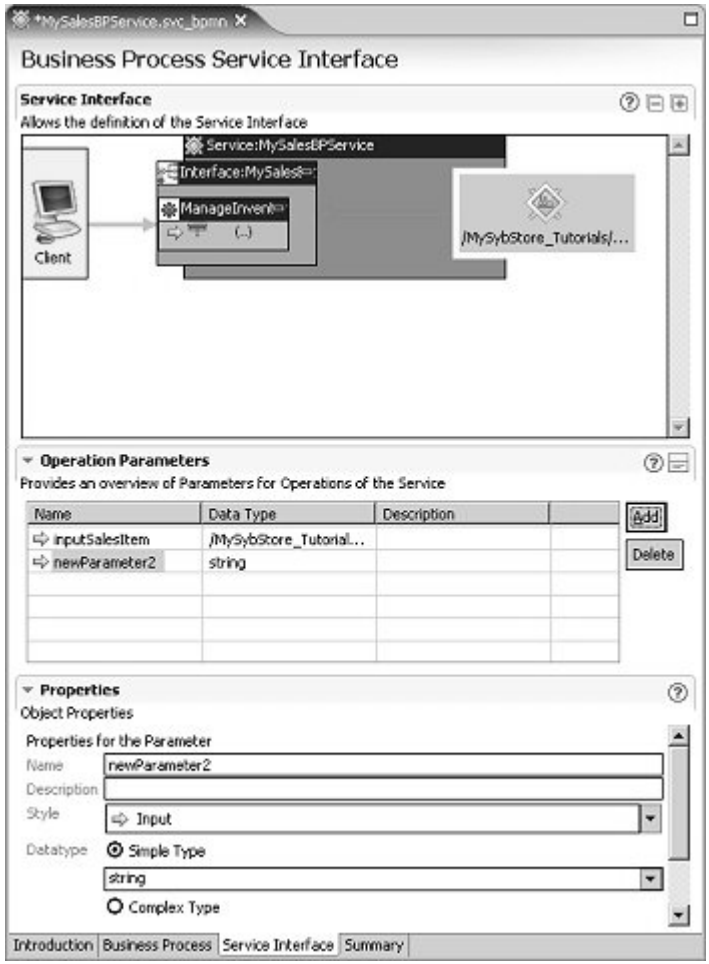


11 Click **Browse for Schema** at the bottom of the **Properties** pane. When the **Schema Browser** opens, expand **MySybStore_Tutorials/Tutorial_Resources/Service_Development** in the tree view, then select the **SybStore.xsd** check box.

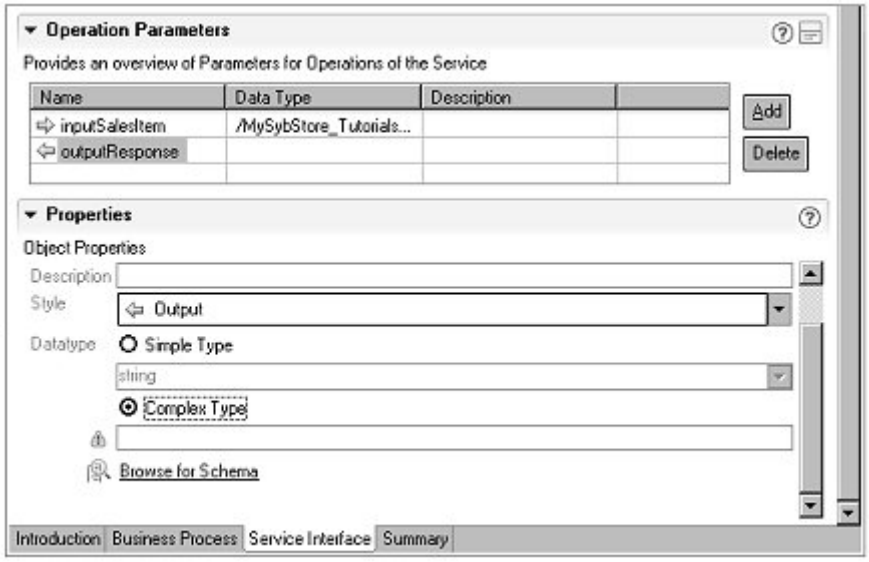
12 In the right pane, select the **SalesDetail** schema at the top of the list and click **OK**.



13 Click **Add** again in the **Operations Parameter** pane to add an output parameter.

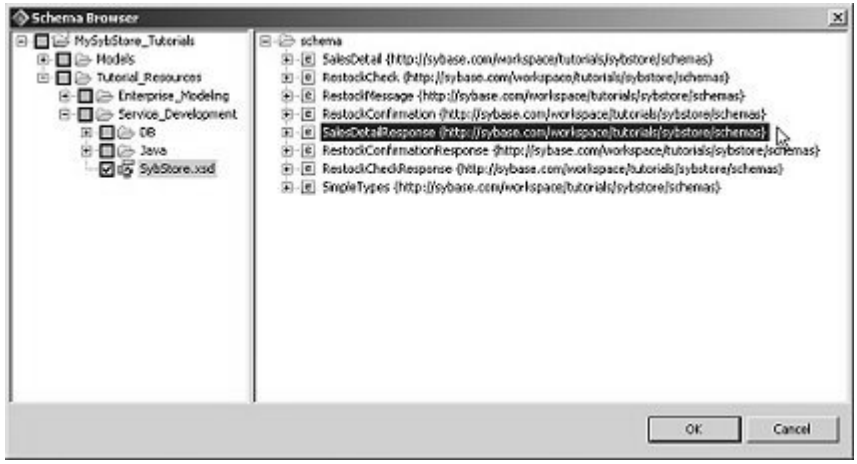


14 In the **Properties** pane, enter `outputResponse` for the **Name**, select **Output** from the **Style** drop-down list, and select **Complex Type** for the **Datatype**.



15 Click **Browse for Schema** in the **Datatype** section. When the **Schema Browser** opens, select **MySybStore_Tutorials/Tutorial_Resources/Service_Development** in the tree view, then select the **SybStore.xsd** check box.

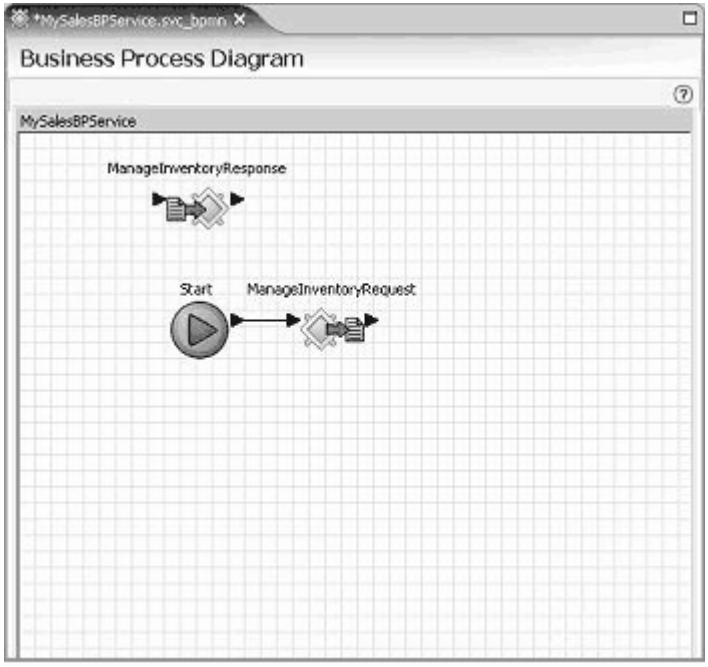
16 In the right pane, select the **SalesDetailResponse** schema and click **OK**.



Note If you select the wrong schema, click **Browse for Schema** in the **Properties** pane to redisplay the schema browser.

17 Select **File|Save** from the WorkSpace main menu.

18 Select the **Business Process** tab in the editor to see the **Business Process Diagram**.



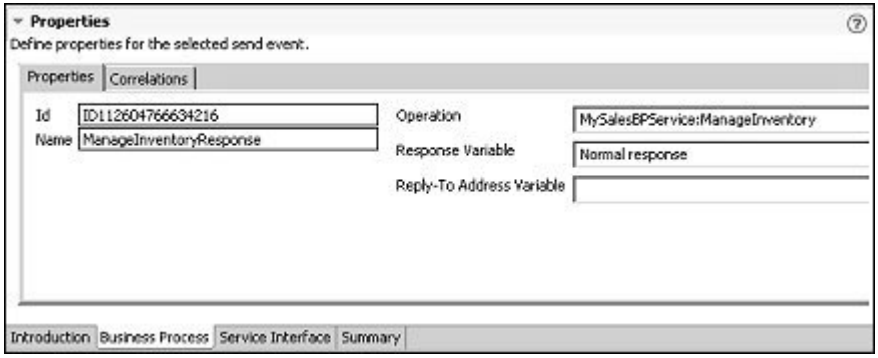
You see two operations—ManageInventoryResponse and ManageInventoryRequest.

Note To change how icon labels display, select **Window|Preferences** on the WorkSpace main menu. When the **Preferences** dialog box opens, select **Sybase, Inc.|Graphic Editors** in the tree view on the left, then make your selections in the **Graphic Editors Icons** section.

19 Expand the **Properties** pane below the **Business Process Variables** pane, then select **ManageInventoryResponse** in the diagram.

Note If you cannot see the entire Properties pane, close the Business Process Variables pane and maximize the editor window.

The ManageInventoryResponse activity is associated with the service operation MySalesBPService:ManageInventory.



So far, the semantics of the business process service is that an invocation of the ManageInventory operation sends the inputSalesItem input parameter to the operation and returns the outputResponse output parameter. For the MySalesBPService:ManageInventory operation to perform a useful function, add activities between the ManageInventoryRequest and ManageInventoryResponse.

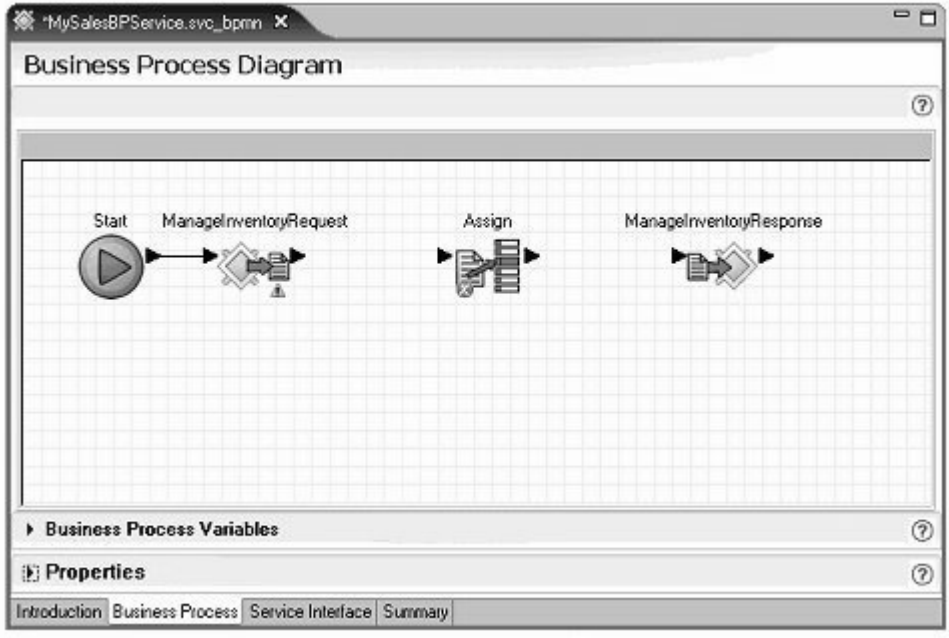
The simplest activity is to assign a value to the *outputResponse* variable before it is returned by the operation. To do this, use an Assign activity.

20 Right-click the background of the diagram and select **Show Tool Palette** from the context menu.

21 On the **Tool Palette**, select the **Assign** activity from the **Activities** category and drag it onto the **Business Process Diagram** canvas.

Note If the Tool Palette disappears from the display, click the Tool Palette icon in the Fast View to redisplay it. You can also right-click the title of the Tool Palette to select various display options. See the online help topic *Sybase WorkSpace Development|Getting Started|Basics|Tool Palettes* for more information.

22 Rearrange the diagram objects to line up horizontally starting with the **Start** icon, followed by the **ManageInventoryRequest** icon to the right, followed by the **Assign** icon, then ending with the **ManageInventoryResponse** icon on the far left.

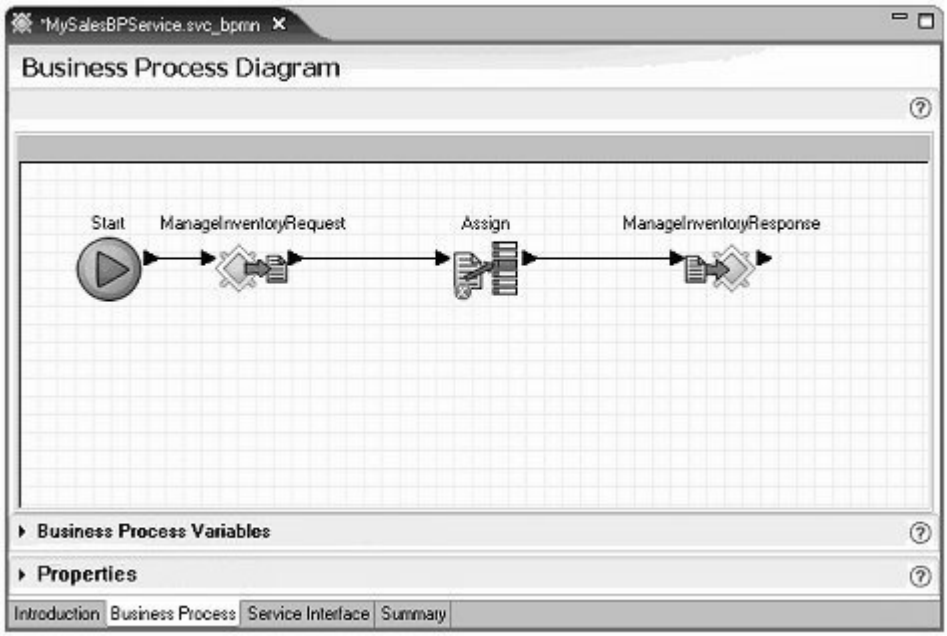


23 Connect the **Assign** activity into the business process flow.

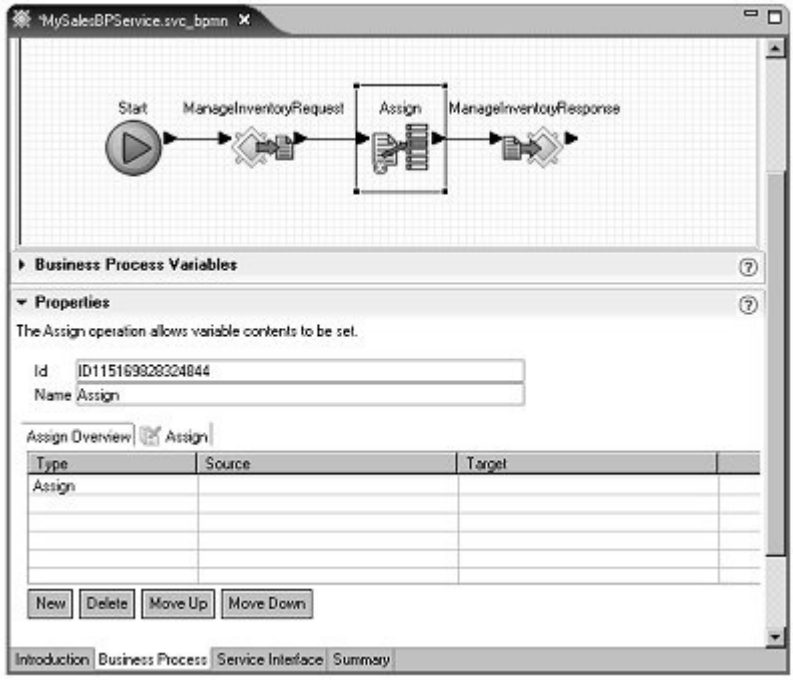
a Click the arrow on the right side of the **ManageInventoryRequest** and drag to and click the arrow on the left side of the **Assign** activity to connect those objects.

b Click the arrow on the right side of the **Assign** activity and drag to and click the arrow on the left side of the **ManageInventoryResponse** target icon to connect those objects.

Your diagram should resemble the following graphic. Notice the red “X” on the Assign activity’s icon. Once you set this activity’s parameters, the “X” disappears.

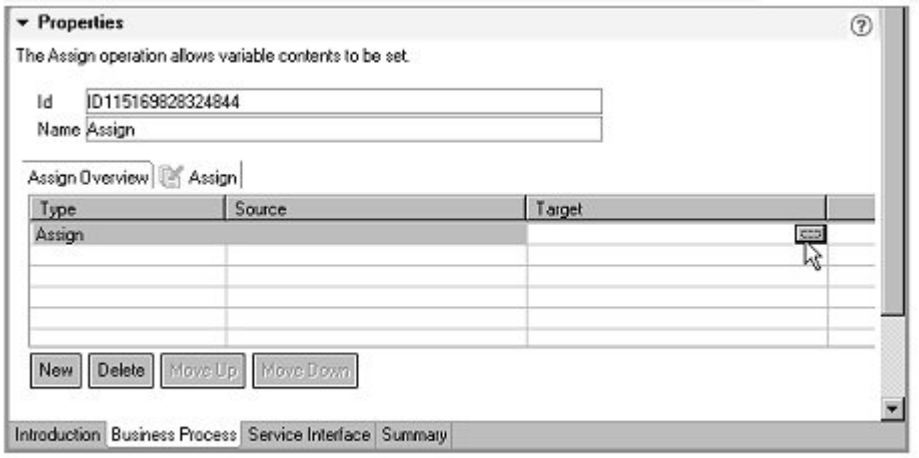


24 Select the **Assign** activity in the diagram and open the **Properties** pane at the bottom of the editor window.

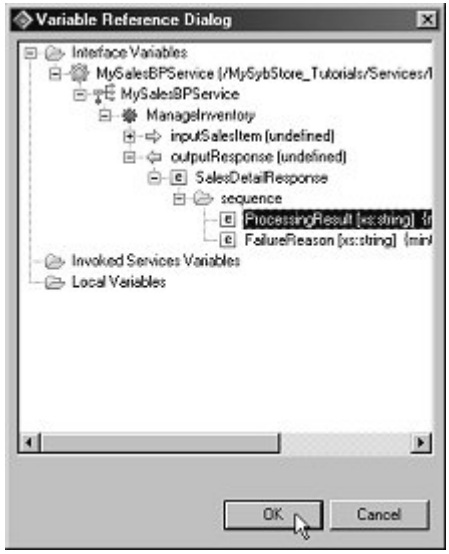


An Assign activity copies a source value to a target variable.

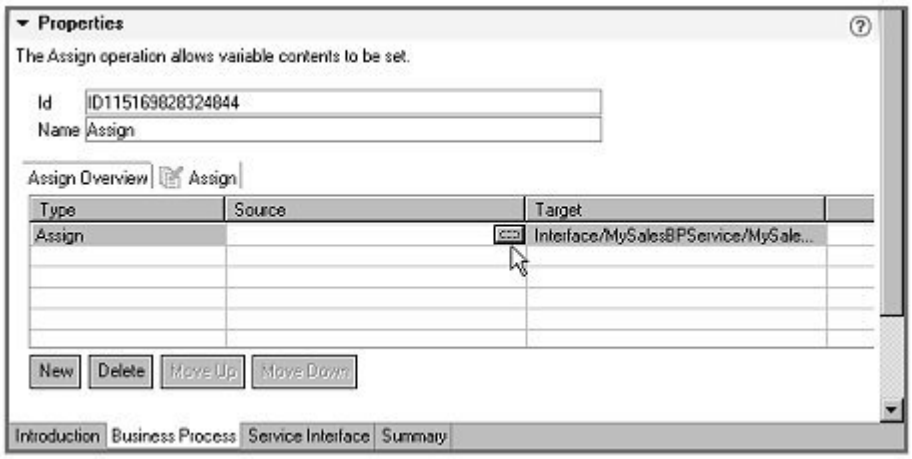
25 In the first row of the table (with `Assign` in the **Type** column), click the cell of the **Target** column, then click the button with the three dots (ellipsis).



26 When the **Variable Reference Dialog** opens, select **Interface Variables/MySalesBPService/MySalesBPService/ManageInventory/outputResponse/SalesDetailResponse/sequence/ProcessingResult** and click **OK**.



27 Again in the first row, click the cell in the **Source** column and click the ellipsis button.



28 When the **Variable Reference Dialog** opens, select the **Literal** option (below the **Variable** pane), type `SUCCESS` in the **Literal** field, and click **OK** to set the value.

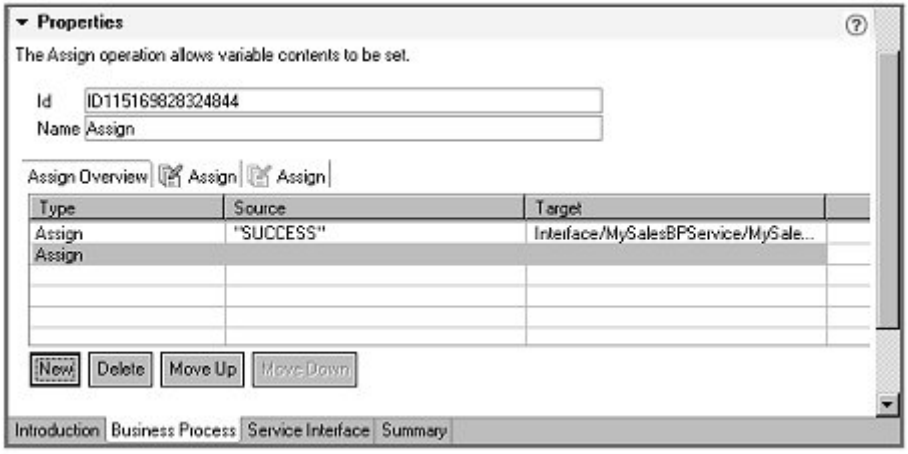
29 Select the **Business Process Variables** pane (above the **Properties** pane), and expand the **Interface Variables/MySalesBPService/MySalesBPService/ManageInventory/outputResponse/SalesDetailResponse/sequence** folder.

The **outputResponse** variable has two sequences:

- **ProcessingResult**
- **FailureReason**

Set a value for **FailureReason**. First, add a new Assign directive to the Assign activity.

30 Close the **Business Process Variables** pane, open the **Properties** pane, then click **New** below the table.



Set the FailureReason part of outputResponse to a literal value.

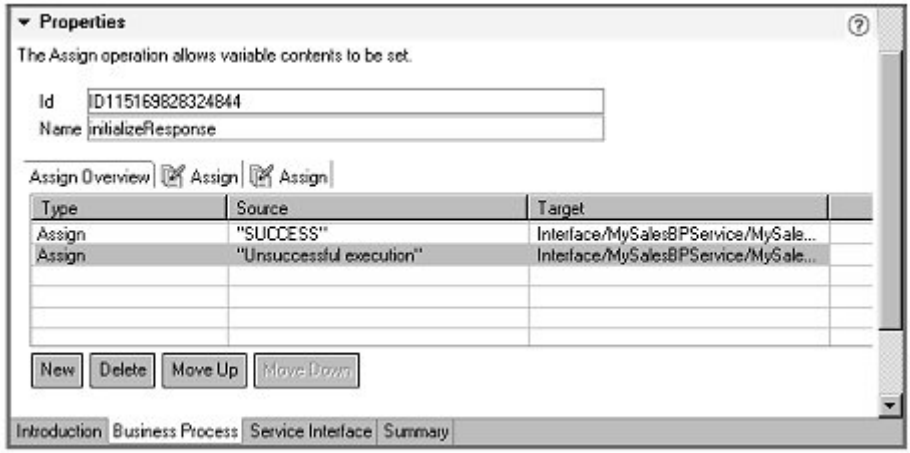
31 In the second row (with `Assign` in the **Type** column), click the cell of the **Target** column, then click the ellipsis button.

32 When the **Variable Reference Dialog** opens, select **Interface Variables/MySalesBPService/MySalesBPService/ManageInventory/outputResponse/SalesDetailResponse/sequence/FailureReason** and click **OK**.

33 Again in the second row, click the cell in the **Source** column, then click the ellipsis button.

34 When the **Variable Reference Dialog** opens, select the **Literal** option (below the **Variable** pane), type `Unsuccessful execution` in the **Literal** field, and click **OK**.

35 Finally, enter `initializeResponse` in the **Name** field.



36 Select **File|Save** to save the business process.

The red "X" no longer displays on the Assign activity's icon in the editor canvas diagram.

You have finished creating the business process service, MySalesBPService, that has one operation named ManageInventory.

The ManageInventory operation takes one input parameter called inputSalesItem, and returns an output parameter called outputResponse. The outputResponse parameter has a complex type with two parts of type string—ProcessingResult and FailureReason.

The ManageInventory operation receives the input parameter, sets the values of the *outputResponse* variable parts to literals, and returns the outputResponse parameter. SalesBPService is a complete service that you can deploy and test.

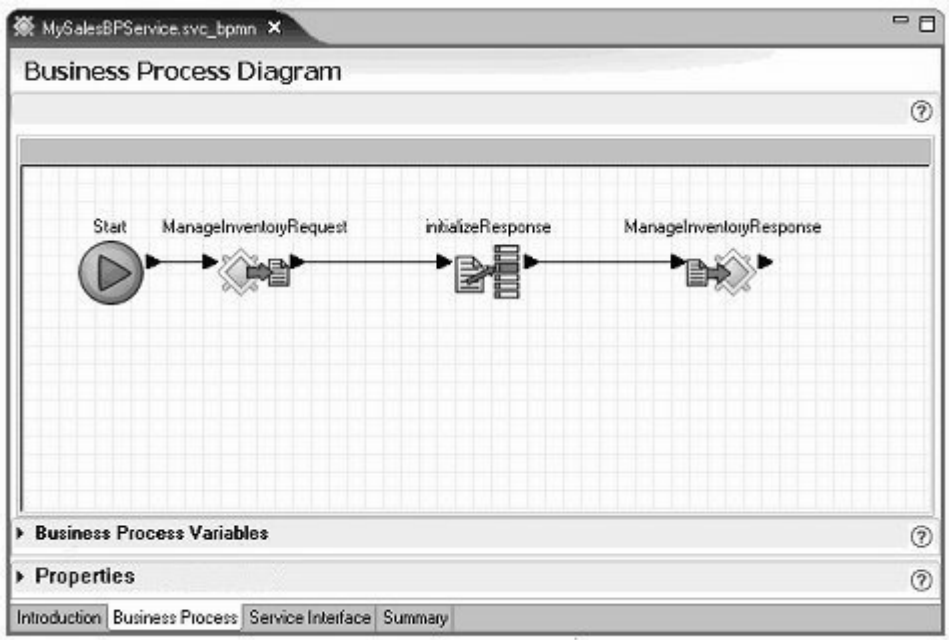
Lesson 2: Adding a service invocation to a business process service

In the first lesson, you created a business process service with a simple operation called ManageInventory. In this lesson, you augment the logic of the ManageInventory operation to invoke a pre-built Java service named SalesValidate.

1 If the business process you created in the previous lesson is not open, in the **WorkSpace Navigator,** expand **MySybStore_Tutorials/Services/BP** and double-click **MySalesBPService.svc_bpmn** to open the file in the Business Process Service Editor.

Add the service invocation inside a complex activity. One of the purposes of a complex activity is to hide details that you do not want to see at the top level of the business process logic.

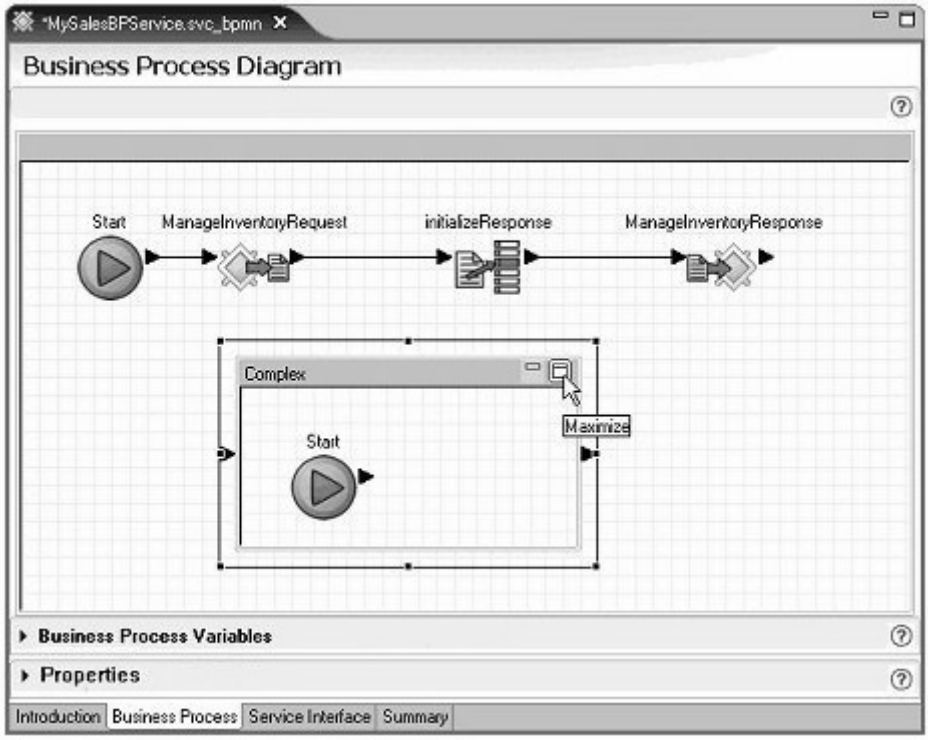
2 Select the **Business Process** tab at the bottom of the editor.



3 When the **Business Process Diagram** displays, right-click anywhere in the editor canvas and select **Show Tool Palette** from the context menu.

4 On the **Tool Palette**, select the **Activities** category, then select the **Complex** activity and drag it onto the editor canvas.

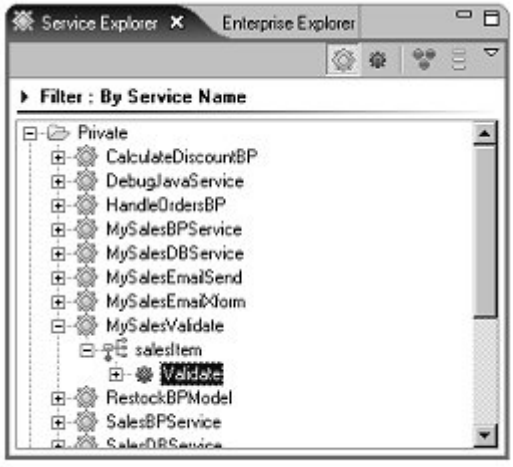
5 Click the maximize icon on the **Complex** activity's title bar.



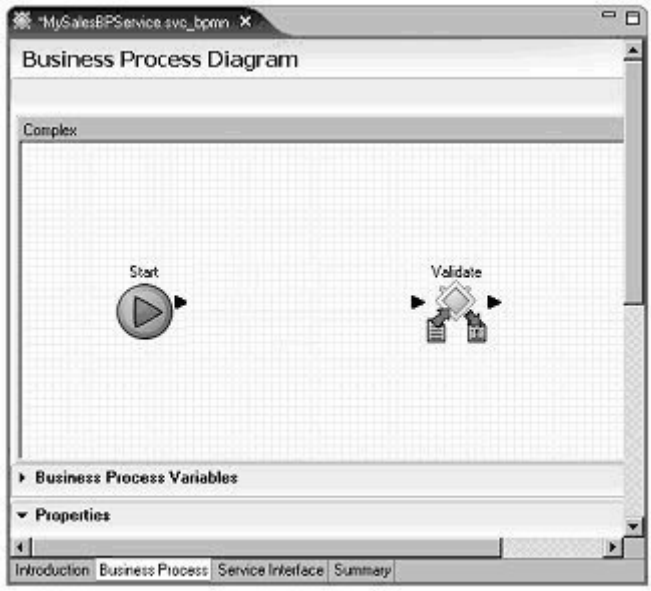
You can also minimize the Business Process Variables and Properties panes to view more of the diagram.

6 Select the **Service Explorer.** This view may be hidden behind the Enterprise Explorer. If you cannot locate it, click **>>1** to the right of the **Enterprise Explorer** tab, then select **Service Explorer** from the list.

7 Expand **Private/MySalesValidate/salesItem**.



8 With the **Complex** activity maximized, drag the **Validate** service from the **Service Explorer** onto the Business Process Diagram canvas.

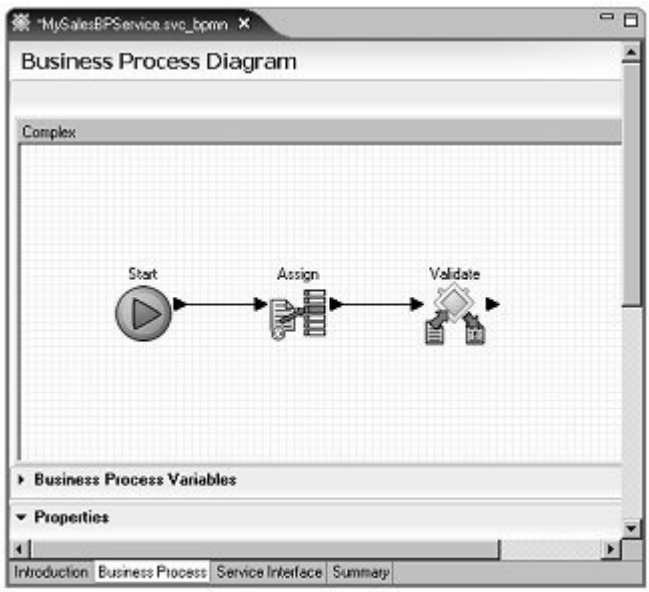


9 Right-click on the canvas and select **Show Tool Palette** from the context menu.

10 Select the **Activities** category on the **Tool Palette** and drag the **Assign** activity onto the canvas.

11 Rearrange the diagram objects horizontally, placing the **Assign** icon between the **Start** icon and the **Validate** icon.

12 Connect the business process logic in the diagram

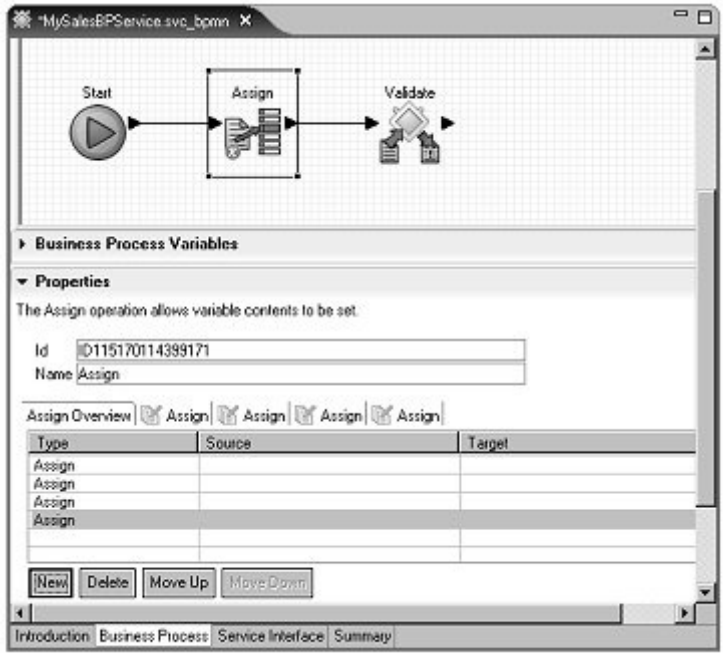


a Click the right arrow on the **Start** icon and drag to and click the left arrow on the **Assign** icon to connect those objects.

b Click the right arrow on the **Assign** icon and drag to and click the left arrow on the **Validate** icon to connect those objects.

13 Select the **Assign** activity in the diagram.

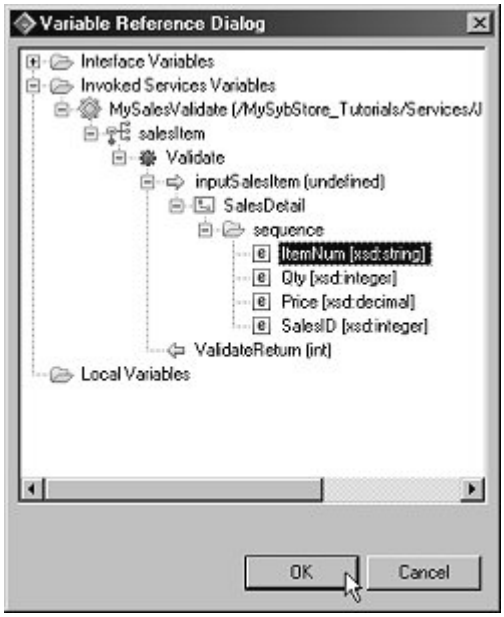
14 Expand the **Properties** pane and click **New** three times below the **Assign Overview** table to create three more directives for the **Assign** activity.



15 Set the target variable for the first `Assign`:

a In the row of the first `Assign` in the **Assign Overview** table, click the cell in the **Target** column, then click the ellipsis button.

b When the **Variable Reference Dialog** window opens, select **Invoked Services Variables/MySalesValidate/SalesItem/Validate/inputSalesItem/SalesDetail/sequence/ItemNum** variable and click **OK**.



16 Repeat the preceding step for each of the other three `Assign` rows in the **Properties** table to set the remaining targets using the *Qty*, *Price*, and *Sales ID* as variables. The other three **Target** columns should reflect these settings:

- Invoked Services Variables/MySalesValidate/SalesItem/Validate/inputSalesItem/SalesDetail/Sequence/Qty
- Invoked Services Variables/MySalesValidate/SalesItem/Validate/inputSalesItem/SalesDetail/Sequence/Price
- Invoked Services Variables/MySalesValidate/SalesItem/Validate/inputSalesItem/SalesDetail/Sequence/SalesID

17 Set the source variable for the first `Assign`:

a In the row of the first `Assign` in the **Assign Overview** table, click the cell in the **Source** column, then click the ellipsis button.

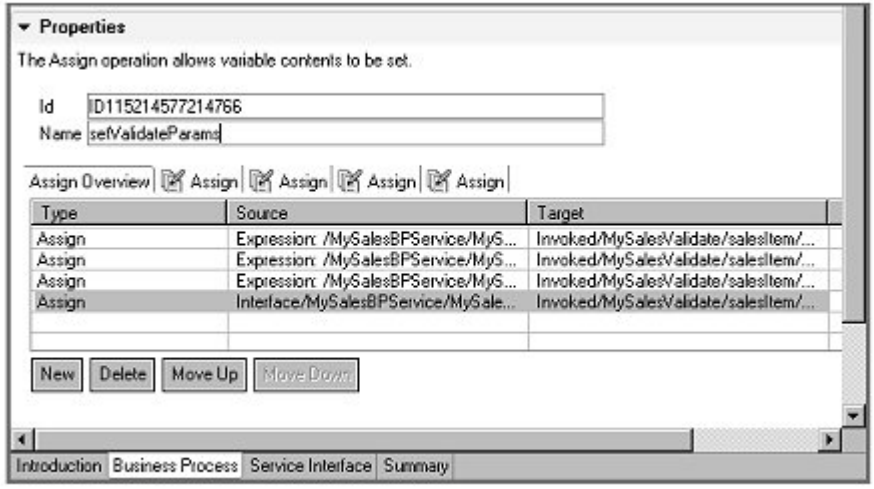
b When the **Variable Reference Dialog** window opens, select **Interface Variables/MySalesBPService/MySalesBPService/ManageInventory/inputSalesItem/SalesDetail/sequence/ItemNum** and click **OK**.

c Repeat steps a and b for the other three Assign Sources. The other three **Source** columns should reflect these settings:

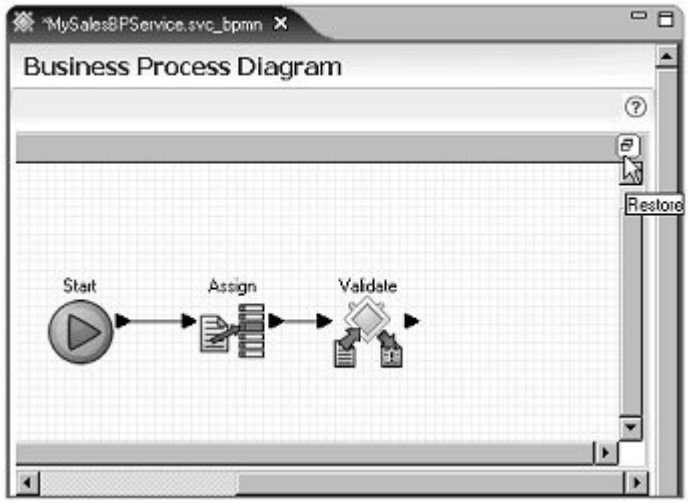
- Interface Variables/MySalesBPService/MySalesBPService/ManageInventory/inputSalesItem/SalesDetail/sequence/Qty
- Interface Variables/MySalesBPService/MySalesBPService/ManageInventory/inputSalesItem/SalesDetail/sequence/Price
- Interface Variables/MySalesBPService/MySalesBPService/ManageInventory/inputSalesItem/SalesDetail/sequence/SalesID

18 In the **Properties** pane, change the name of the **Assign** activity by entering `setValidateParams` in the **Name** field.

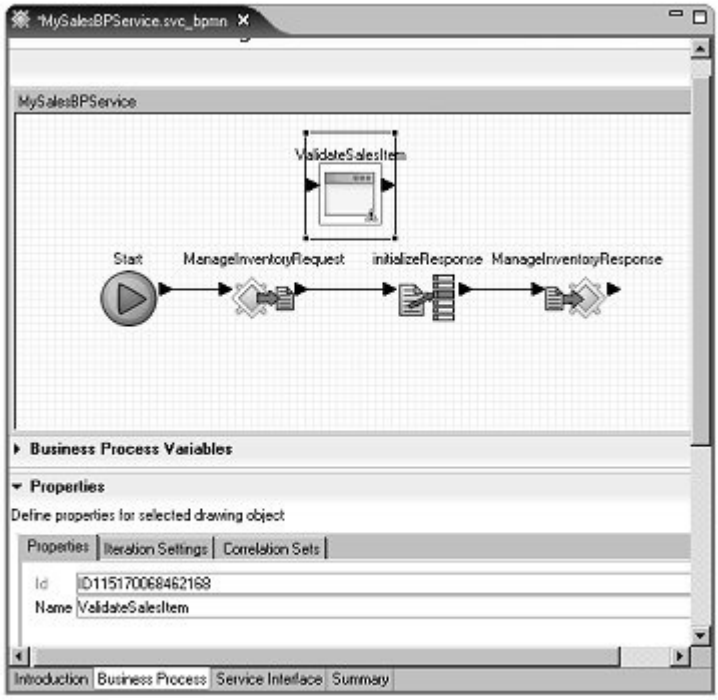
When you finish, the Business Process Variables section of the editor should look like this:



19 Restore the **Complex** activity (click the small double window icon in the upper-right corner of the Complex activity diagram) to return to the top-level of the business process diagram, then minimize the **Complex** activity small window to make the icon the same as the other objects within the top-level diagram.



20 Select the **Complex** activity icon in the Business Process Diagram, expand the **Properties** pane below the diagram, and rename the **Complex** activity by entering `ValidateSalesItem` in the **Name** field.



21 Add the ValidateSalesItem activity between the initializeResponse and ManageInventoryResponse activities in the business process flow:

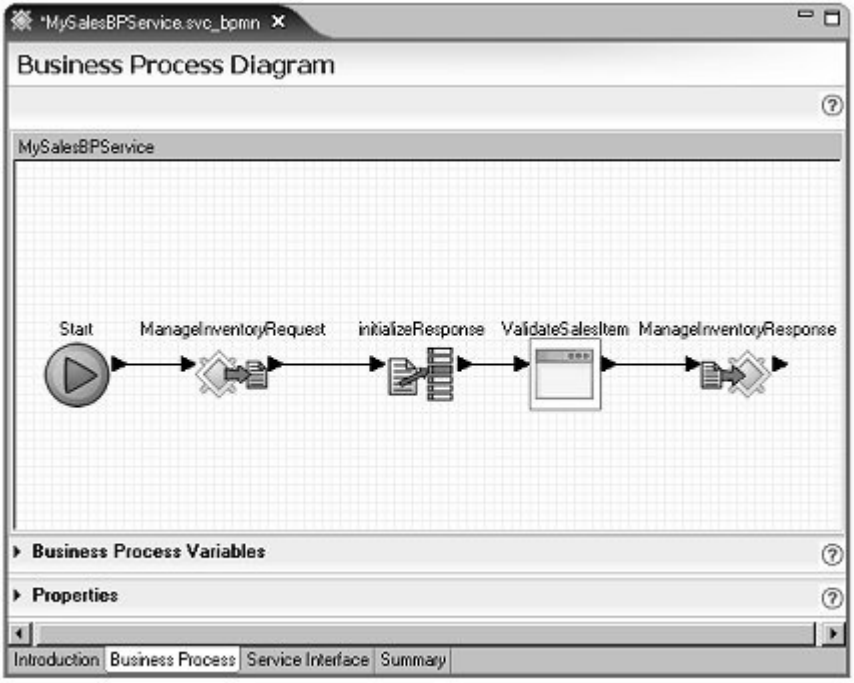
a Right-click the connector line between **initializeResponse** and the **ManageInventoryResponse** in the diagram and select **Delete** from the context menu.

b Drag the **ValidateSalesItem** icon in between the **initializeResponse** icon and the **ManageInventoryResponse** icon. Rearrange the icons to fit the diagram as necessary.

c Click the right arrow of the **initializeResponse** icon and drag and click the left arrow of the **ValidateSalesItem** icon to connect those activities.

d Click the right arrow of the **ValidateSalesItem** icon and click the left arrow of the **ManageInventoryResponse** icon to connect those activities.

Your business process flow looks like the following graphic with Start connected to ManageInventoryRequest, connected to initializeResponse, connected to ValidateSalesItem, connected to ManageInventoryResponse.



22 Select **File|Save** from the WorkSpace main menu bar.

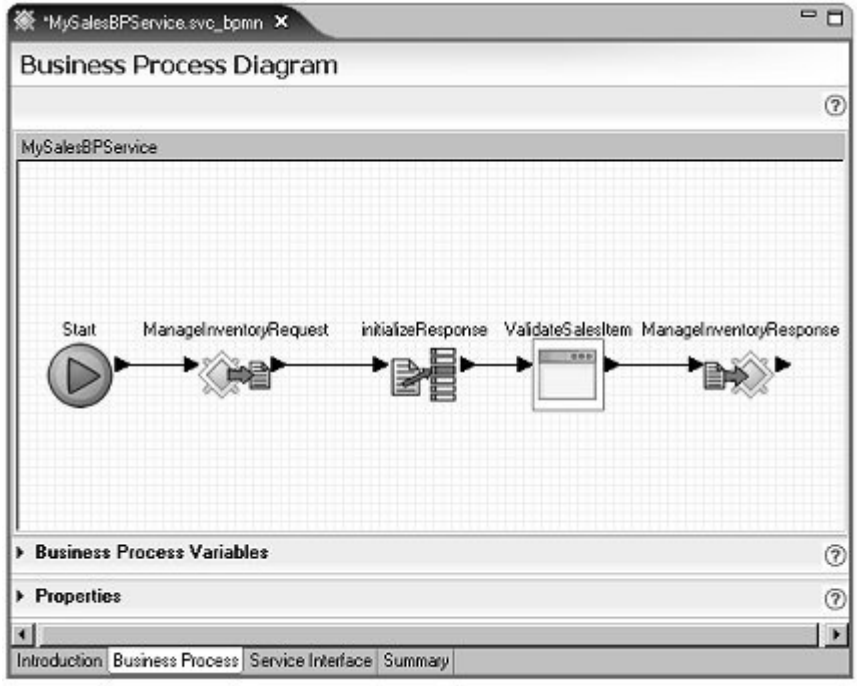
You have finished adding a service invocation to your business process logic.

The ManageInventory operation now takes one input parameter inputSalesItem, validates the sales item data with a Java service, and returns an output parameter outputResponse.

Lesson 3: Adding a rule to a business process service

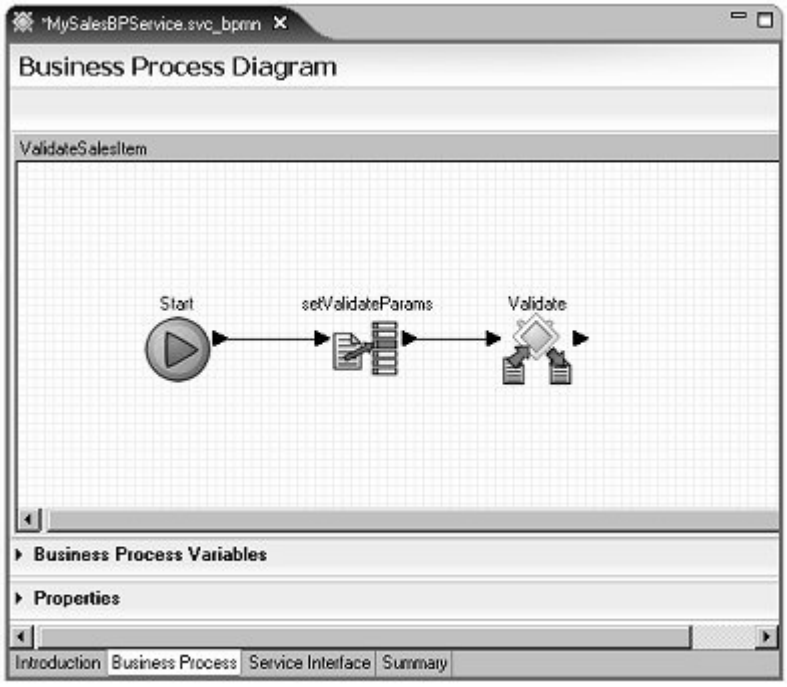
In this lesson, you add a rule to check the results of the service invocation you created in the previous lessons.

1 If **MySalesBPService.svc_bpmn** is not open, in the **WorkSpace Navigator**, expand **MySybStore_Tutorials/Services/BP** and double-click **MySalesBPService.svc_bpmn** to open it in the Business Process Service Editor.



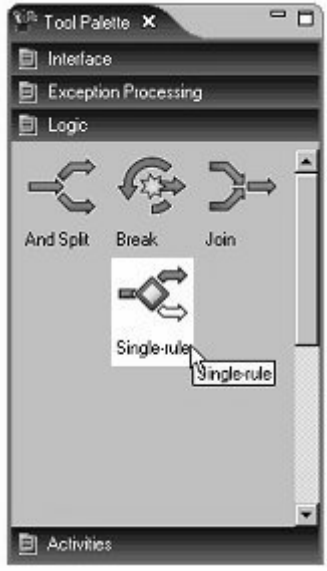
2 Select the **Business Process** tab in the editor.

3 Double-click **ValidateSalesItem** in the diagram to open the activity to open it, then click **ValidateSalesItem** maximize icon to expand the display.



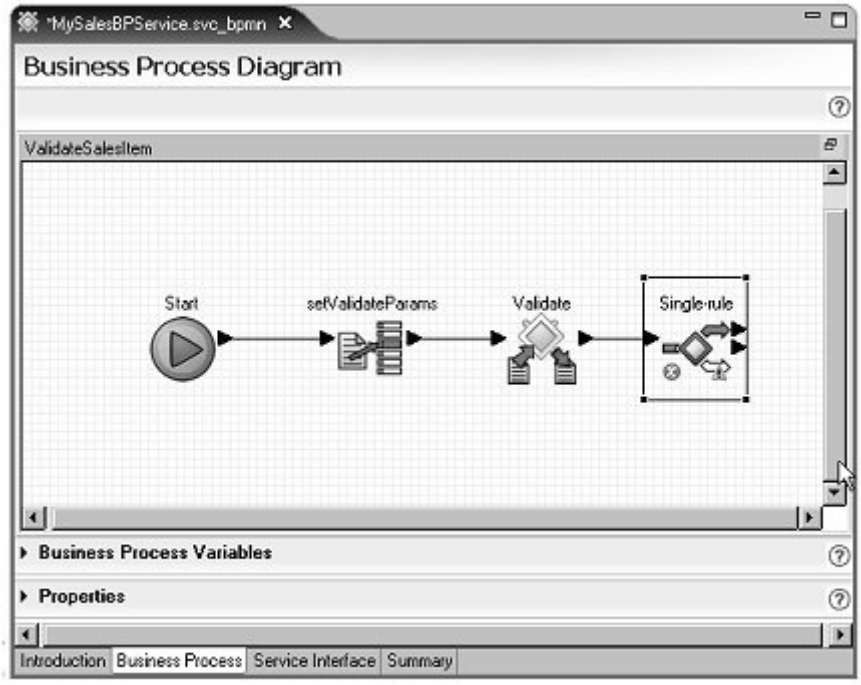
Add single-rule logic to the business process logic.

4 Right-click in the editor diagram and select **Show Tool Palette**.



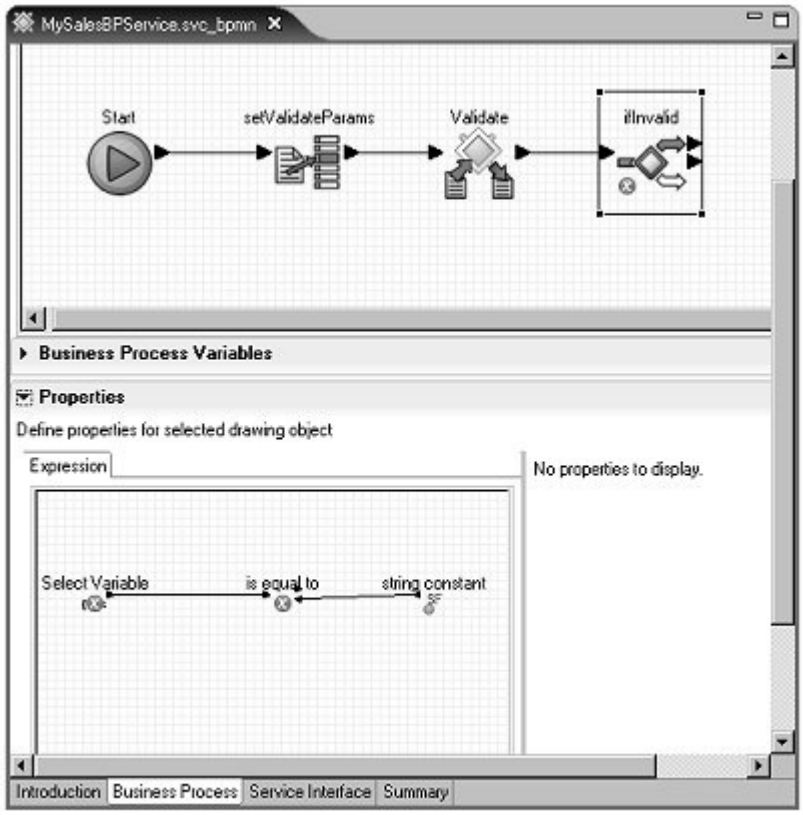
5 On the **Tool Palette**, select the **Logic** category, then drag the **Single-Rule** logic onto the canvas to the right of the **Validate** icon.

6 Connect these icons in the business process logic. Click the right arrow of the **Validate** activity and drag to and click the left arrow of the **Single-rule** activity.



7 Rename the **Single-rule** activity by selecting the icon on the canvas, double-clicking the name and entering `ifInvalid`.

8 Select the **ifInvalid** activity on the canvas and expand the **Properties** pane at the bottom of the editor window. The ifInvalid logic displays in the Expression editor.



Note To expand the Expression editor window, click the right border of the window and drag out.

The expression you want to build for the ifInvalid rule is:

```
ValidateReturn is not equal to 1
```

9 Expand the **Business Process Variables** pane and select **Invoked Services Variables/MySalesValidate/SalesItem/Validate/ValidateReturn**.

10 Drag **ValidateReturn** from the **Business Process Variables** pane on top of the **Select Variable** icon in the **Expression** editor in the **Properties** pane.

11 Select the **is equal to** operator in the **Expression** editor. The properties for the operator display to the right of the Expression editor.

12 Change the **is equal to** operator properties. Select **!=** from the **Source Text** drop-down list. The **Logical Name** automatically changes to `is not equal to`.

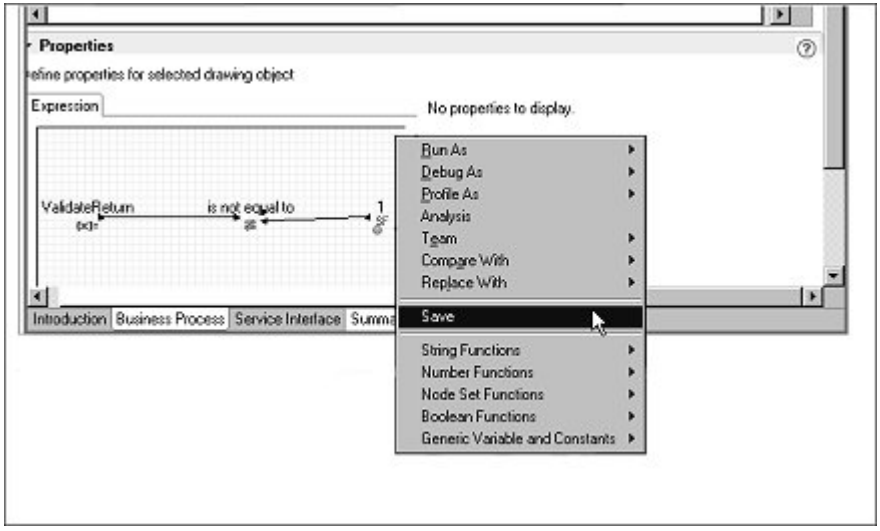
Note A red "X" displays for this operator in the Expression editor. The "X" changes to the proper "is not equal to" sign when you save your changes.

---

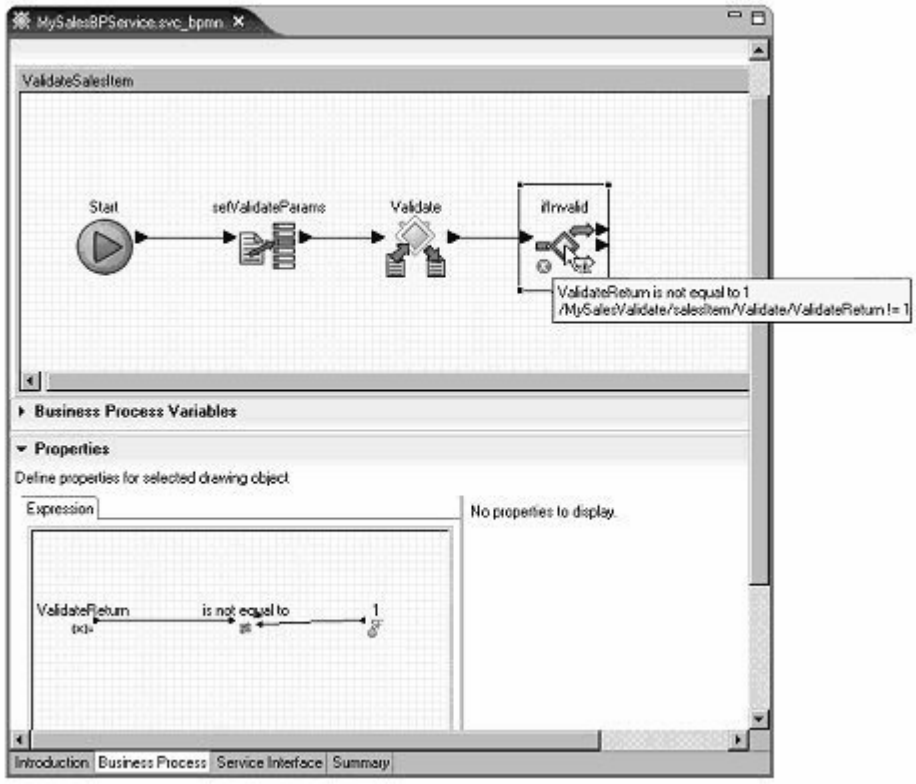
13 Select **string constant** in the **Expression** editor.

14 Change the **Logical Name** to `1`, select the **Bind Source Text to Logical Name** option, and select **int** from the **Data Type** drop-down list. The value in the **Source Text** field changes to "`1`".

15 Right-click the background of the **Expression** editor canvas and select **Save** from the context menu.



16 Move the cursor over the **ifInvalid** activity on the business process canvas to see the rule expression.



17 Minimize the **Properties** pane, but leave the ValidateSalesItem activity maximized in the editor diagram.

The ifInvalid single rule continues to be marked with a red "X" until you map one of its paths in a later lesson. You have finished adding a rule to a business process service to check the results of a service invocation.

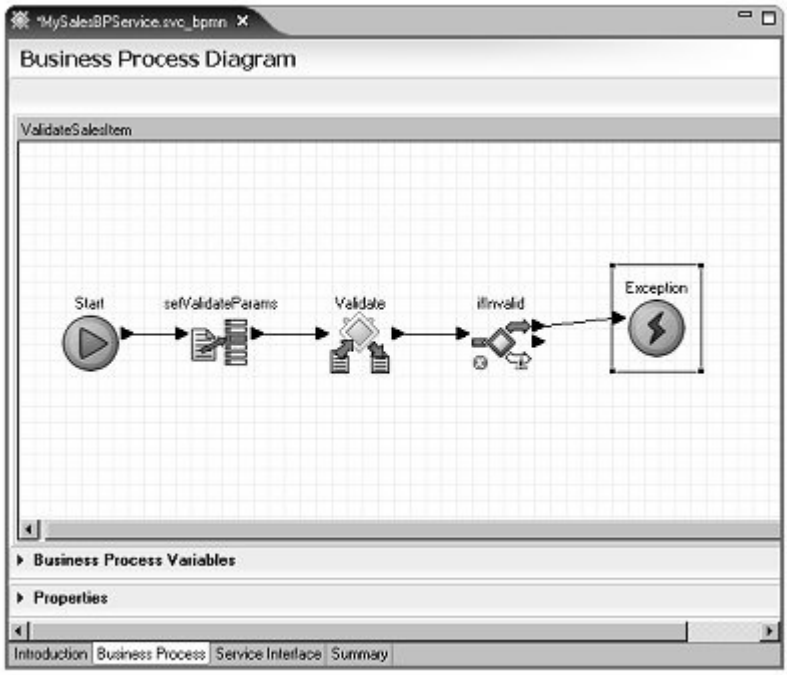
Lesson 4: Defining error handling for a business process service

In this lesson, you process an invalid sales item by throwing an exception and handling that exception.

1 Right-click the background of the business process canvas and select **Show Tool Palette**.

2 In the **Tool Palette**, select the **Exception Processing** category, then drag the **Throw Exception** activity to the right of the **ifInvalid** icon on the business process canvas.

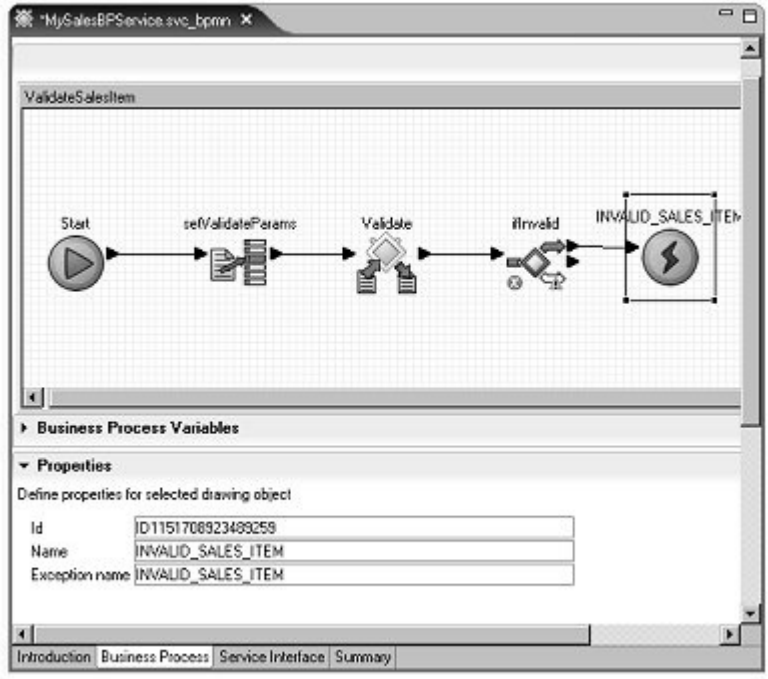
3 Connect the logic. Click the right top arrow (TRUE) of the **ifInvalid** single-rule activity and drag and click the left arrow of the **Exception** icon on the canvas.



Note The red "X" on the ifInvalid object disappears when you save the your changes in a later step.

4 Select the **Exception** icon on the canvas and expand the **Properties** pane.

5 Enter `INVALID_SALES_ITEM` in the **Name** and **Exception Name** field in the **Properties** pane.

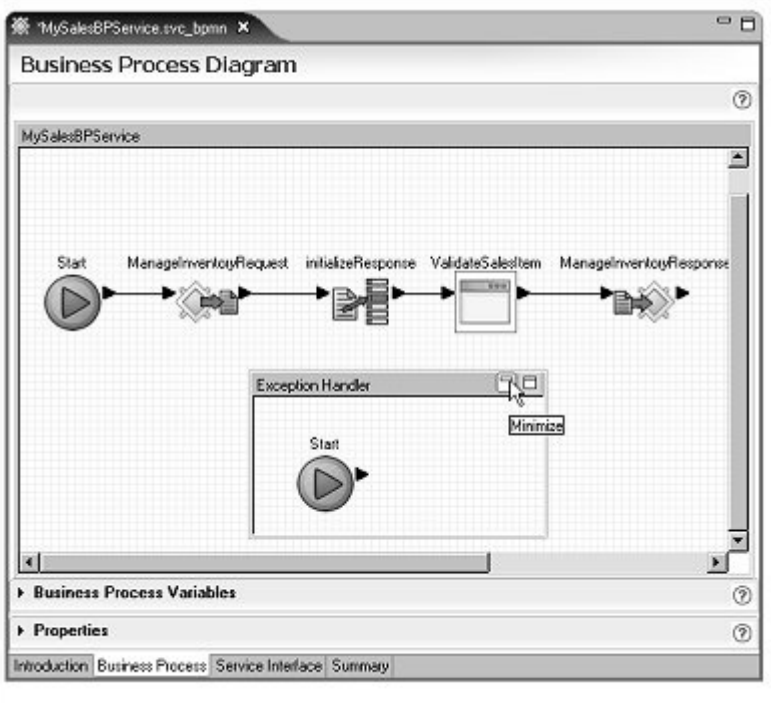


6 Right-click on the canvas and select **Save** from the context menu to save the changes.

7 Restore (minimize) the **ValidateSalesItem** complex activity, minimize it, right-click the background of the business process canvas, and select **Show Tool Palette**.

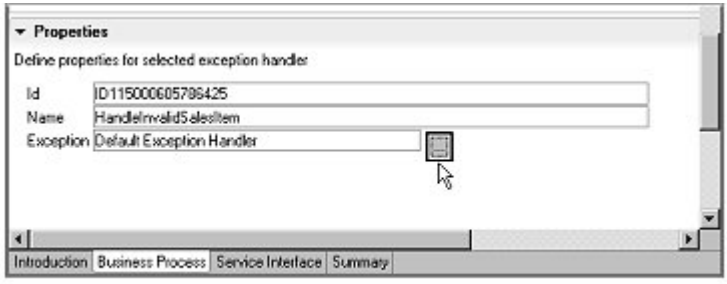
8 On the **Tool Palette**, select the **Exception Processing** category, select the **Exception Handler** activity, and drag and drop it to the editor canvas.

9 The **Exception Handler** displays semi-maximized. Click the **Exception Handler** minimize icon.



10 Select the **Exception Handler** icon on the canvas, expand the **Properties** pane, and enter `HandleInvalidSalesItem` in the **Name** field.

11 In the **Properties** pane, click the ellipsis button next to the **Exception** field to open the **Exception Selection Dialog** window.



12 Expand **ValidateSalesItem**, select `INVALID_SALES_ITEM`, and click **OK**.

13 Select **File|Save** from the WorkSpace main menu.

14 In the editor canvas, double-click the **HandleInvalidSalesItem** activity to expand it, then click its maximize icon to open up the window.

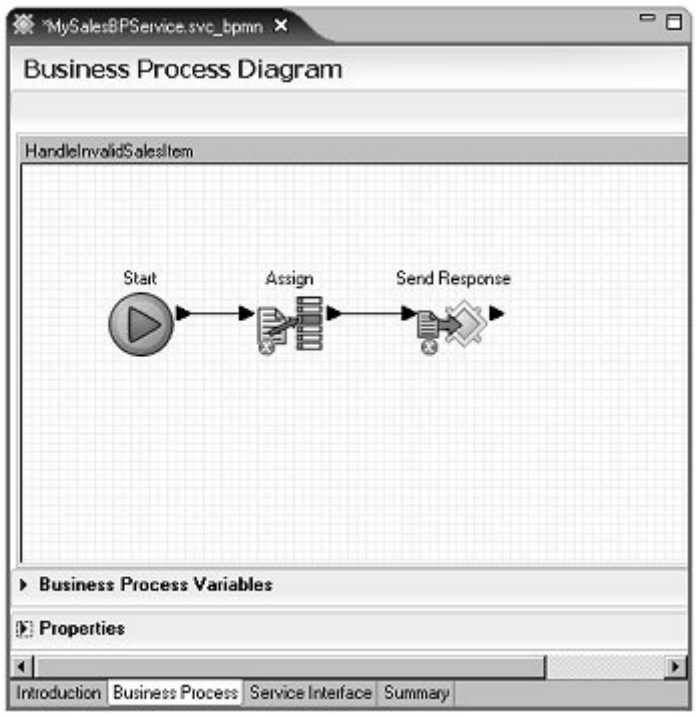
15 Right-click in the editor canvas and select **Show Tool Palette** from the context menu. Add two activities:

   a Select the **Activities** category and drag the **Assign** activity onto the canvas to the right of the **Start** icon on the canvas.

   b Select the **Interface** category and drag the **Send Response** activity onto the canvas to the right of the **Assign** activity.

16 Connect the activities:

   - Click the right arrow on the **Start** activity and drag it to and click the input arrow on the left of the **Assign** activity.
   - Click the output arrow on the right of the **Assign** activity and drag it to and click the input arrow on the left of the **Send Response** activity.

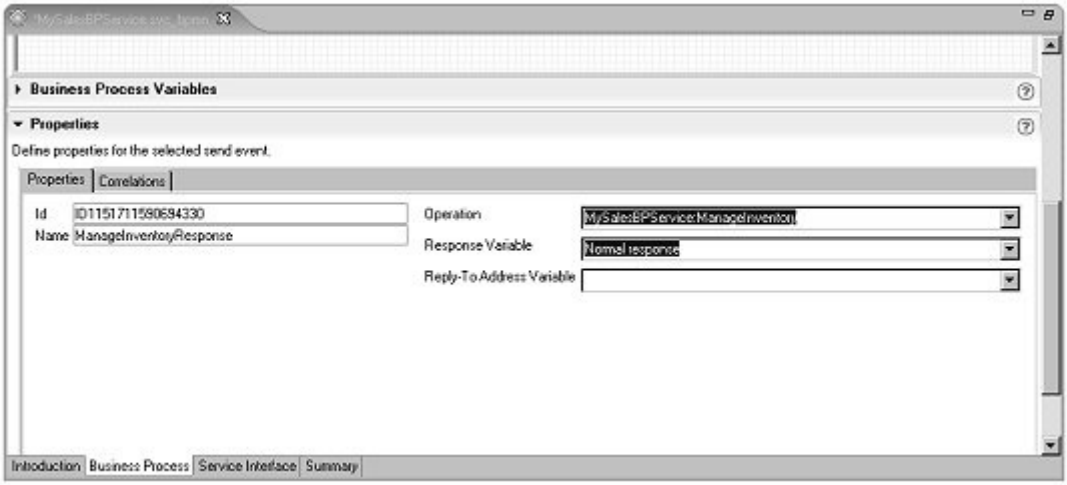


17 Select the **Assign** activity on the canvas, expand the **Properties** pane, and enter `setErrorInfo` in the **Name** field.

18 Select the **SendResponse** activity on the canvas and set these values in the **Properties** pane:

- Name – enter `ManageInventoryResponse`.
- Operation – select `MySalesBPService:ManageInventory` from the drop-down list.
- Response Variable – verify that `NormalResponse` is selected from the drop-down list.

Note You may need to maximize the WorkSpace window to see the arrow for the drop-down lists.



19 Select the **setErrorInfo** activity on the canvas. In the **Properties** pane, click **New** to add a second **Assign** in the **Assign Overview** table.

20 Set the **Assign** values in the **Assign Overview** table in the **Properties** pane:

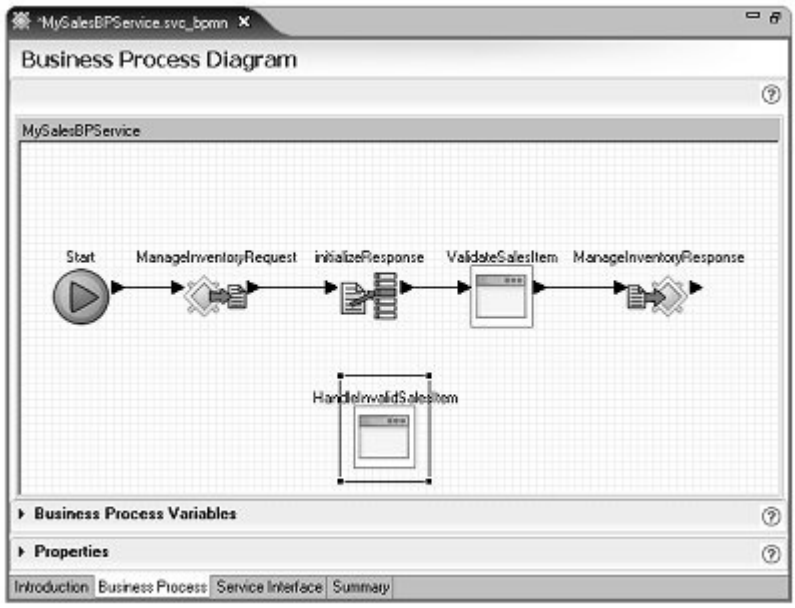
- First Assign – click the ellipsis button in the **Source** column, select **Literal**, enter `FAILURE` in the **Variable Reference Dialog,** and click **OK**.

  Click the ellipsis button in the **Target** column, select `Interface Variables/MySalesBPService/MySalesBPService/ManageInventory/outputResponse/SalesDetailResponse/sequence/ProcessingResult`, and click **OK**.

- Second Assign – click the ellipsis button in the **Source** column, select **Literal**, enter `INVALID SALES ITEM` in the **Variable Reference Dialog,** and click **OK.**

  Click the ellipsis button in the **Target** column, select `Interface Variables/MySalesBPService/MySalesBPService/ManageInventory/outputResponse/SalesDetailResponse/sequence/FailureReason`, and click **OK**.

21 Restore and minimize the **HandleInvalidSalesItem** activity in the editor canvas.



22 Select **File|Save** from the WorkSpace main menu.

Note To take a break, select **File|Close** from the WorkSpace main menu to close the editor, and select **File|Exit** to shut down WorkSpace.

You have finished adding the raising and handling of an exception to a business process service.

The business process service you created has an operation called ManageInventory, which takes a sales item as input and validates the sales item data with a Java service. If the sales item is valid, the operation returns a response indicating success. Otherwise, it returns a response indicating that the sales item is invalid.

Lesson 5: Setting message context properties dynamically

This lesson teaches you how to use message context to send out-of-band data as part of an invocation. For example, you can use message context to dynamically set properties, such as correlation set IDs, when a message service operation is invoked.

You set the subject of an e-mail message sent using the Email service. This lesson assumes that you are somewhat familiar with service deployment and testing from other tutorials in this guide.

1 Select **Window|Open Perspective|Other**, select **Service Development**, and click **OK**.

Add message context to the business process service that requires using some predefined schemas, which are packaged as a project template.

2 Select **File|New|Project** from the WorkSpace main menu bar.

3 When the **New Project** wizard opens, select **Sybase|New Project From Template** in the **Wizards** list, then click **Next**.

4 When the **Select a Project Template** window opens, select **WorkSpace Project with Schemas** and click **Finish**.

5 In the **WorkSpace Navigator**, expand the new project folder **WorkSpaceProjectWithSchemas/Schemas**.

6 Right-click **emailHeader.xsd** and select **Copy** from the context menu.

7 Right-click the **MySybStore_Tutorials/Schemas** folder and select **Paste**.

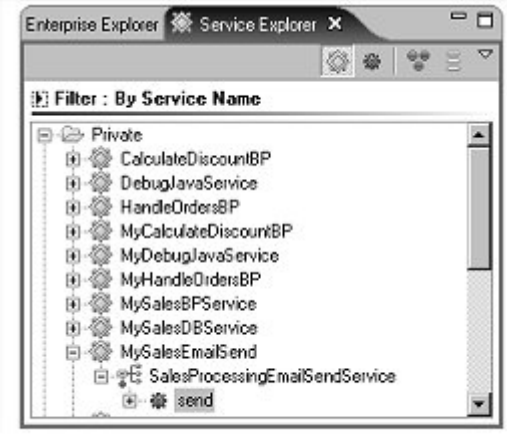
8 If **MySalesBPService** is not open in the editor, expand **MySybStore_Tutorials/Services/BP/** in the **WorkSpace Navigator**, double-click **MySalesBPService.svc_bpmn** to open it in the Business Process Service Editor, and select the **Business Process** tab.

9 Right-click in the Business Process Diagram canvas and select **Show Tool Palette** from the context menu.

Note If the Tool Palette disappears from view, click the Tool Palette icon in the Fast View to redisplay it.

10 Select the **Activities** category, then drag and drop an **Assign** activity onto the diagram to the right of **ManageInventoryResponse**.

11 In the **Service Explorer** view, expand **Private/MySalesEmailSend/SalesProcessingEmailSendService,** select the **send** service and drag and drop it onto the business process canvas to the right of the **Assign** activity you just added.

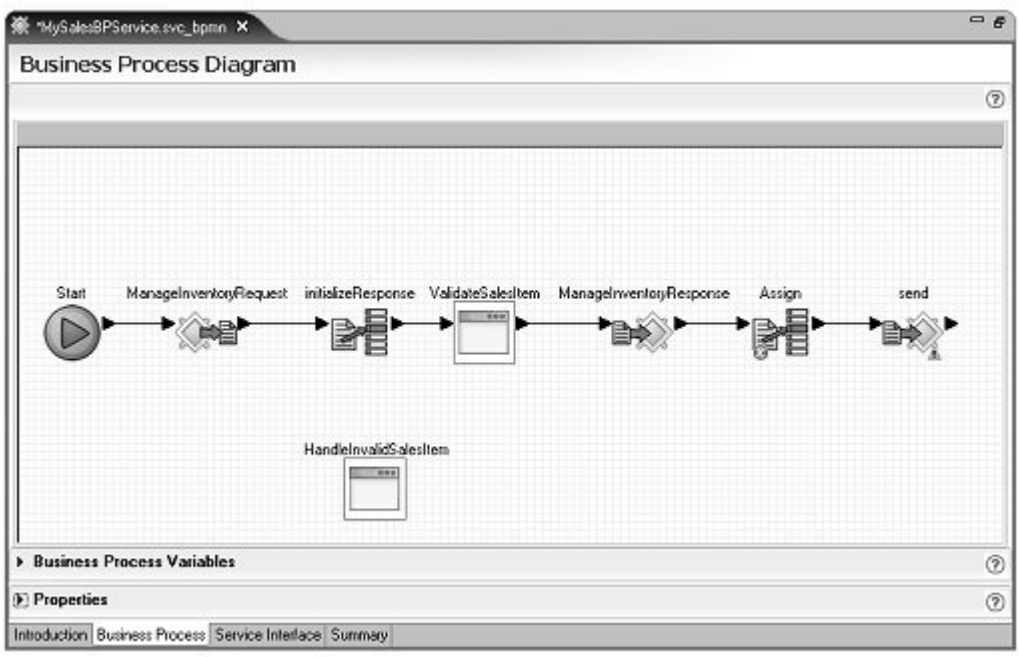


12 Connect the **Assign** and **send** activities into the business process flow. Rearrange the objects on the canvas as necessary to connect the activities.

a Click the arrow on the right side of the **ManageInventoryResponse** and drag to and click the arrow on the left side of the **Assign** activity to connect those icons.

b Click the arrow on the right side of the **Assign** activity and drag to and click the arrow on the left side of the **send** target icon to connect those icons.

The canvas should look like this:

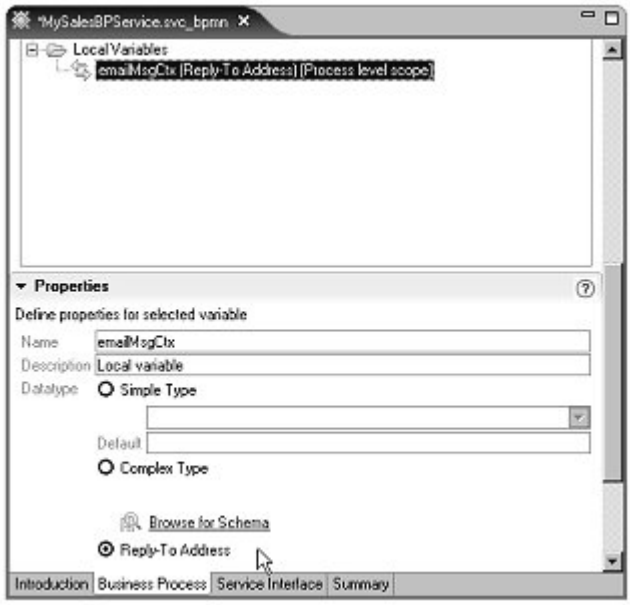


13 Expand the **Business Process Variables** pane in the editor (below the canvas).

14 To create a new local variable, right-click the **Local Variables** folder and select **New Variable** from the context menu.

15 In the **Business Process Variables** pane, expand the **Local Variables** folder and select **newVariable1**.

16 Expand the **Properties** pane, enter `emailMsgCtx` in the **Name** field and select **Reply-To Address** for the **Datatype**.



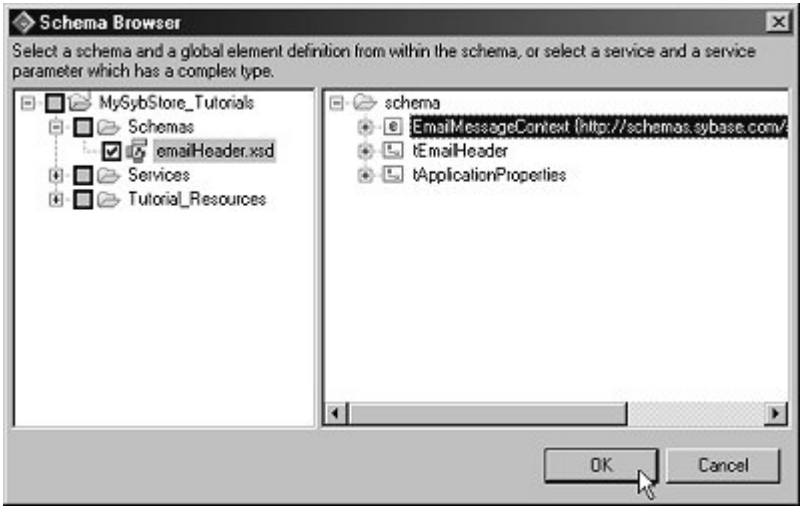
17 Right-click the **Local Variables** folder and select **New Variable** from the context menu to create another new local variable.

18 In the **Business Process Variables** pane, expand the **Local Variables** folder and select **newVariable1**.

19 In the **Properties** pane, complete these properties:

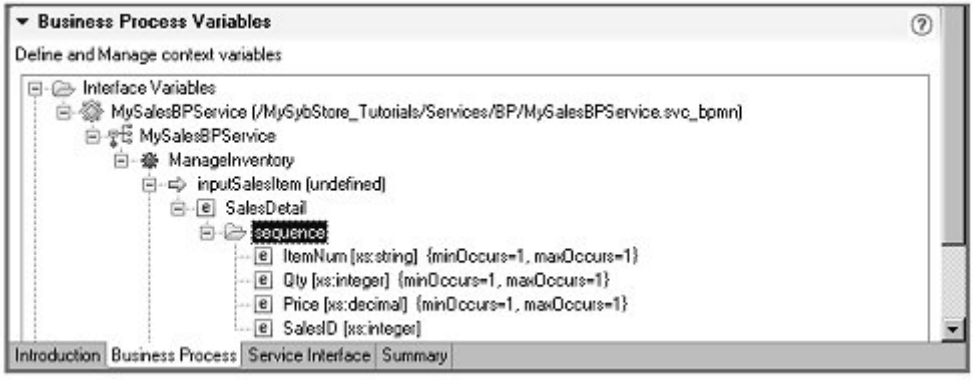
- Name – enter `emailMsgVar`.
- Datatype – select **Complex Type** and click **Browse for Schema**. When the **Schema Browser** opens, expand **MySybStore_Tutorials/Schemas**, and select the **emailHeader.xsd** check box.

20 In the **Schema Browser** right pane, select **EmailMessageContext** below the **schema** folder and click **OK**.



21 Expand the **Properties** pane, then select the new **Assign** icon on the editor canvas.

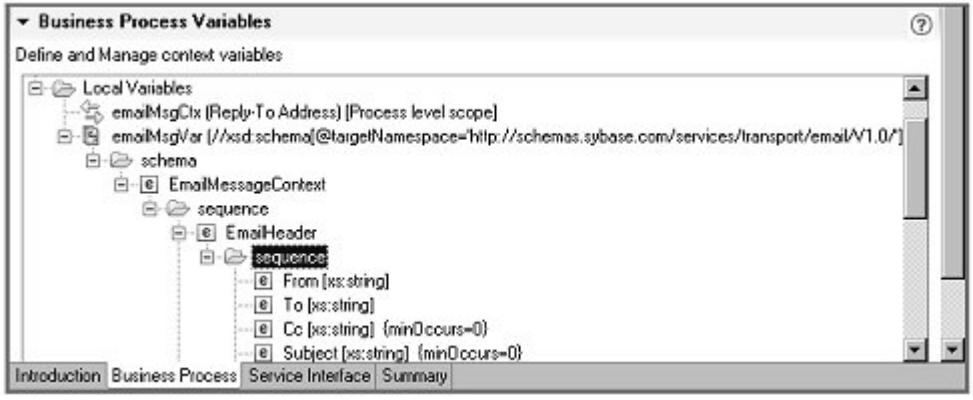
22 Expand the **Business Process Variables** pane, then expand **Interface Variables/MySalesBPService/MySalesBPService/ManageInventory/inputSalesItem/SalesDetail/sequence**.



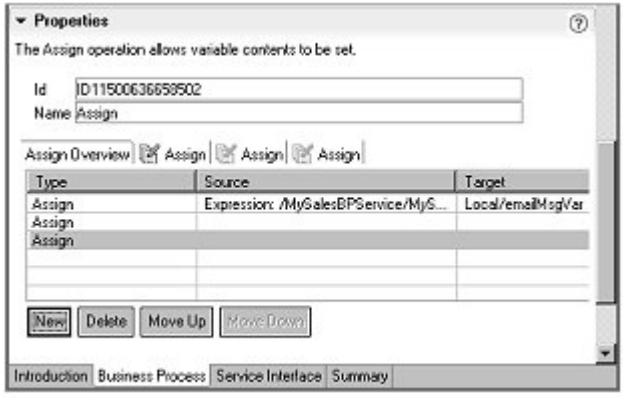
23 Drag and drop **ItemNum** from the **Business Process Variables** pane into the **Source** column of the **Assign** in the **Properties** pane **Assign Overview** table.

24 In the **Business Process Variables** pane, expand **Local Variables/emailMsgVar/schema/EmailMessageContext/sequence/EmailHeader/sequence**.

25 Drag and drop **Subject** onto the **Target** column of the **Assign** variable in the **Properties** pane.

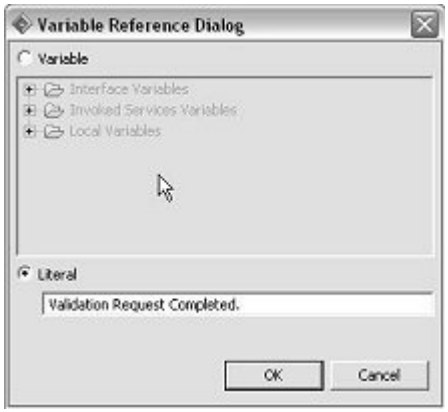


26 In the **Properties** pane, click **New** twice to add two additional **Assign** variables in the **Assign Overview** table.



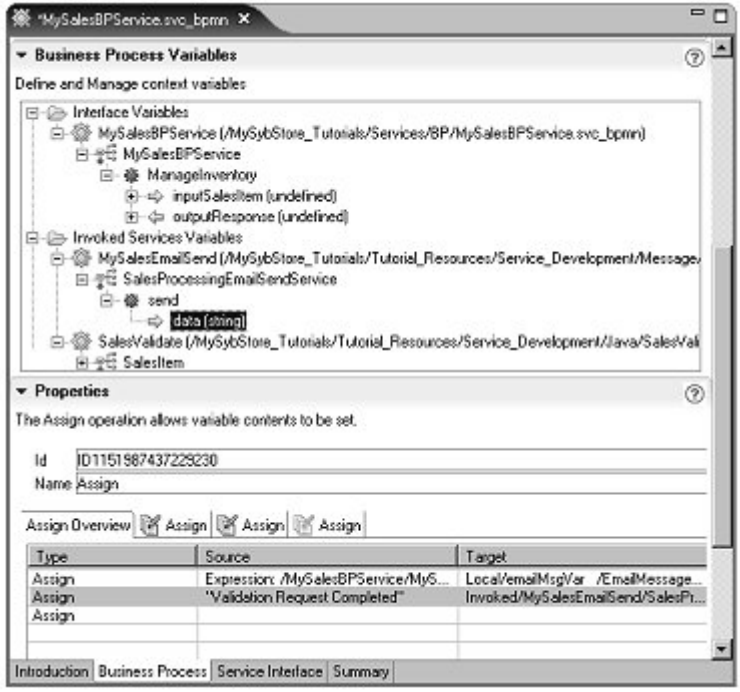
27 Select the **Source** column of the second **Assign** variable and click the ellipsis button.

28 In the **Variable Reference Dialog**, select the **Literal** option, enter `Validation Request Completed`, and click **OK**.



29 In the **Business Process Variables** pane, expand **Invoked Service Variables/MySalesEmailSend/SalesProcessingEmailSendService/send**.

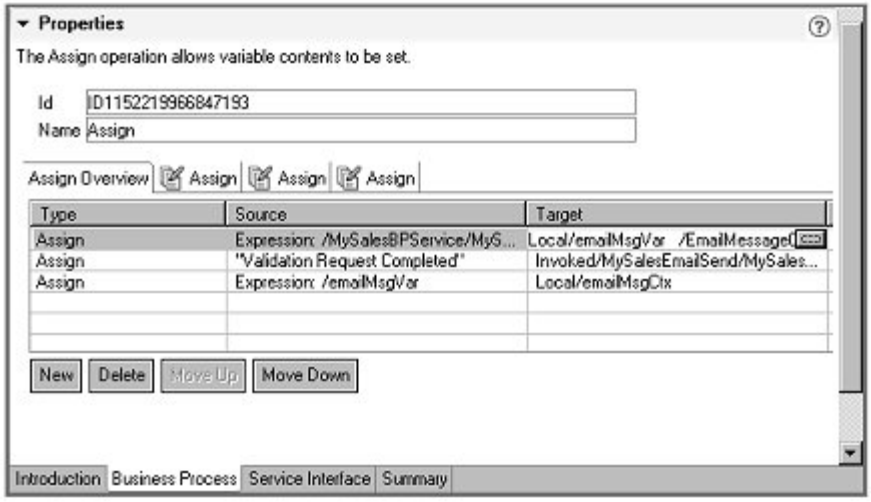
30 Drag and drop **data (string)** from the **Business Process Variables** pane to the **Target** column of the second **Assign** variable in the **Properties** pane.



31 For the third **Assign** variable, in the **Business Process Variables** pane, expand **Local Variables** and drag and drop **emailMsgVar** to the **Source** column in the **Properties** pane.

32 For the **Target** column of the third **Assign** variable, in the **Business Process Variables** pane, expand **Local Variables** and drag and drop **emailMsgCtx** to the **Target** column in the **Properties** pane.

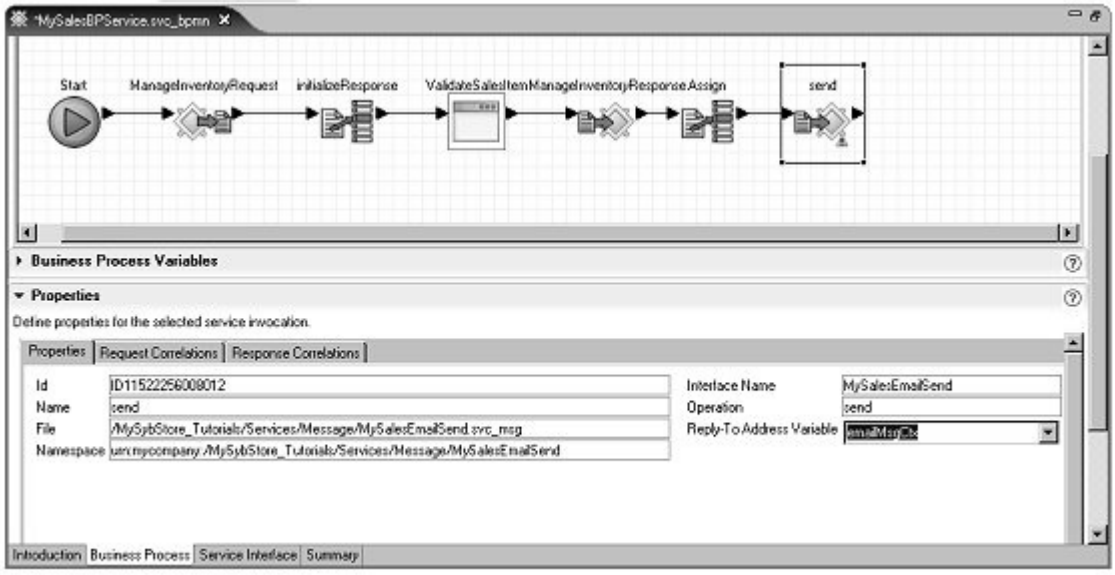
The finished **Assign Overview** table should look like this:



33 In the **Business Process Diagram** canvas, select the **send** operation operation.

34 In the **Properties** pane **Reply-To AddressVariable** field (to the right of the **File** field), select **emailMsgCtx** from the drop-down list.

Warning! Do not type the variable name in the Reply-To Address Variable field; you must select the name from the field's drop-down list. If this field does not display in its entirety on your screen, temporarily change your screen resolution to 1280 x 1024.



35 Select **File|Save** from the WorkSpace main menu bar, then select **File|Close**.

Packaging, deploying, and testing a business process service

This tutorial give you the opportunity to package, deploy, and test the business process service that you created in the previous tutorial.

The service that you test—MySalesBPService—includes an operation that sends an e-mail message. Therefore, before you build the package, you create the e-mail configuration that allows the e-mail message to be sent and received successfully for your test.

This tutorial consists of:

Lesson 1: Building the package
Lesson 2: Deploying the package
Lesson 3: Testing the service

Lesson 1: Building the package

This lesson teaches you how to build a package profile and a package for a business process service. The package profile contains a description of the package and is used when the actual package is built.

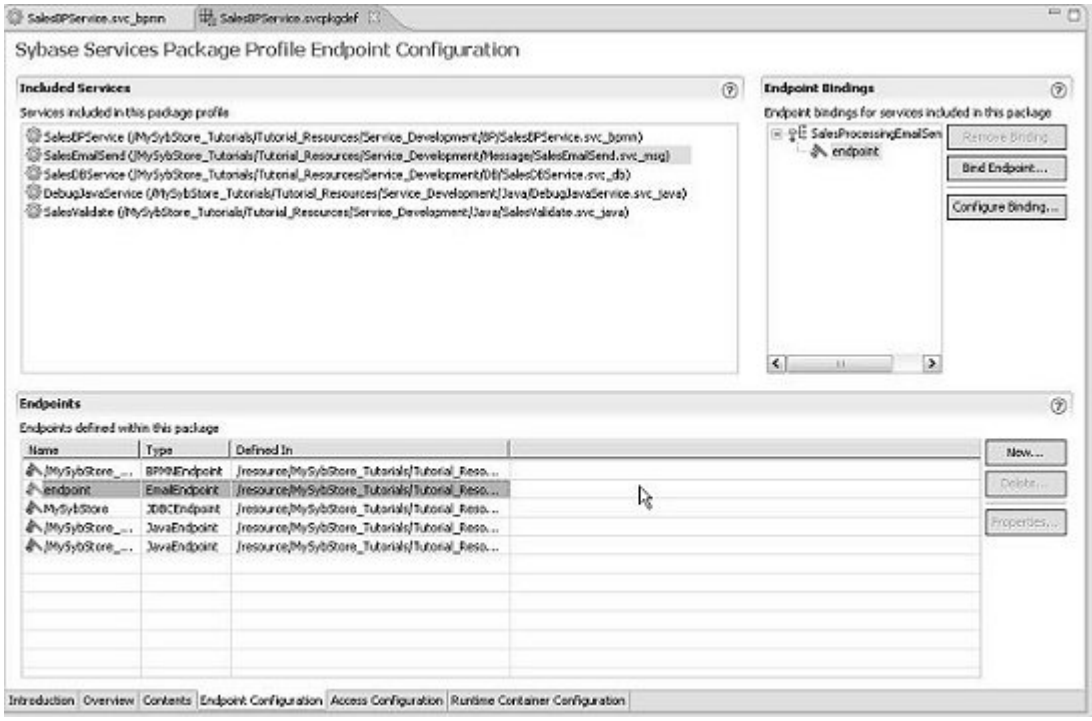
1 Open the **Service Development** perspective in WorkSpace.

2 In the **WorkSpace Navigator**, expand **MySybStore_Tutorials/Services/BP**, right-click **MySalesBPService.svc_bpmn**, and select **Create Sybase Services Package Profile** from the context menu.

The new package profile—*MySalesBPService.svcpkgdef*—is created and opens in the Sybase Services Package Profile Editor.

To run the business process in the WorkSpace test environment, configure the e-mail endpoint and the e-mail host.

3 In the editor, select the **Endpoint Configuration** tab and click **New** to the right of the table in the **Endpoints** pane.



4 When the **New Endpoint Wizard** opens, select **Messaging Endpoint**, and click **Next**.

5 In the **Endpoint Name** window, enter `myemailendpoint` for the **Name** and click **Next**.

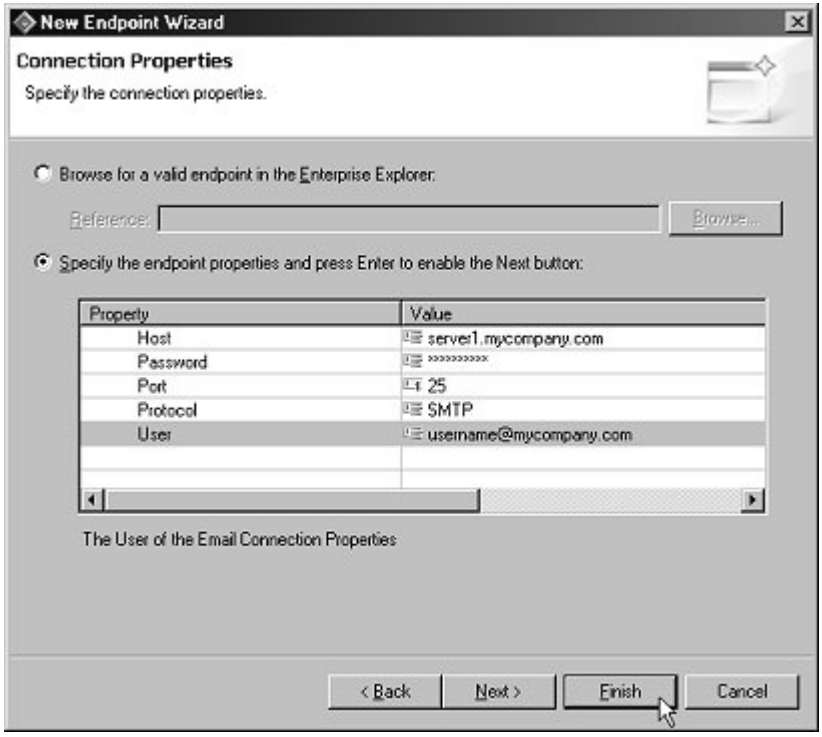
6 In the **Messaging Type** window, select **Email** and click **Next**.

7 In the **Connection Properties** window, select **Specify the Endpoint Properties ....** option and set the **Host**, **Password, Port**, and **User** to the appropriate values for your e-mail; for example:

- Host – `machinename.yourcompany.com` (the host of your e-mail server)
- Password – `******` (your password to access the host, which displays as asterisks)
- Port – `25` (the default port used to access the host)

- User – `rob.thomas@yourcompany.com` (your e-mail address)

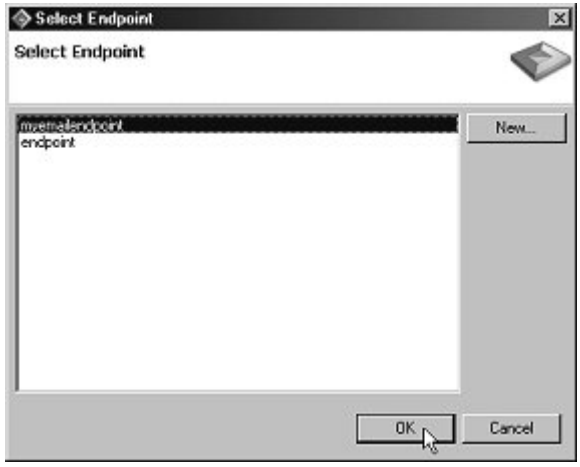
8 Click **Finish**.



9 In the **Included Services** section of the **Endpoint Configuration** page, select **SalesEmailSend**.

10 In the **Endpoint Bindings** section (to the right of **Included Services**), highlight **SalesProcessingEmailSendService/endpoint** and click **Bind Endpoint**.

11 In the **Select Endpoint** dialog box, select **myemailendpoint** and click **OK**.



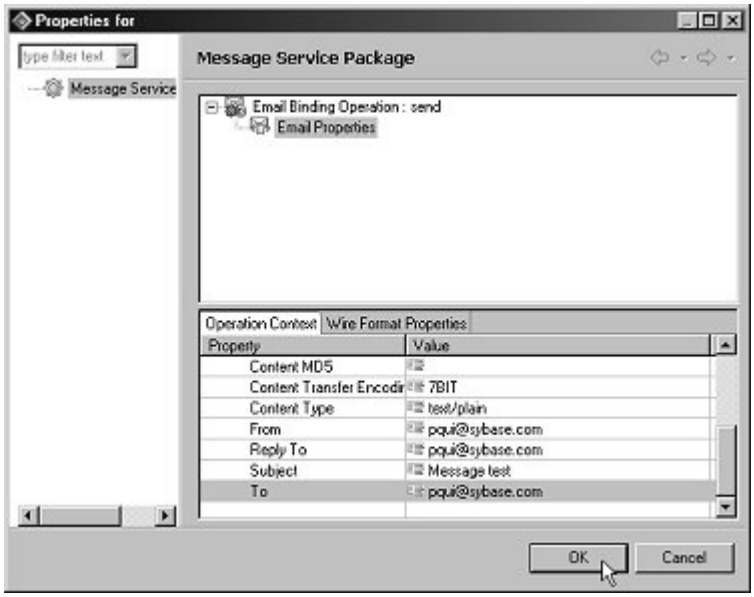
12 In **Endpoints Bindings** section, select **SalesProcessingEmailSendService/myemailendpoint** and click **Configure Binding**.

13 When the **Properties** dialog box opens, select **Email Binding Operation:send/Email Properties** in the top right pane, and select the **Operation Context** tab in the bottom right pane.

14 In the **Operation Context** pane, scroll down and set the **From**, **Reply To, Subject**, and **To** properties, as appropriate for your environment, then click **OK**.

For example, if your e-mail address is rob.thomas@youcompany.com, you would enter that in the From, Reply To, and To fields, and enter `My Email Test` in the Subject field.

Note The From, Reply To, and To values must be in the form of an e-mail address; for example, rob.thomas@sybase.com.



15 Select **File|Save** from the WorkSpace main menu bar, then select **File|Close**.

16 In the **WorkSpace Navigator**, right-click **MySalesBPService.svcpkgdef** and select **Build Package** from the context menu. This builds the actual package from the package profile you created earlier in the lesson.

Note If a message prompts whether you want to overwrite an existing package file, click **Yes to All**.

The Console view opens and shows the progress of the operation.

17 When a message states that the package was successfully created, click **OK**.

Lesson 2: Deploying the package

Before you test the package, deploy it to the Unwired Orchestrator server.

1 Using the Windows taskbar, select **Start|Programs|Sybase|Sybase WorkSpace|UO 5.1|Start UO**.

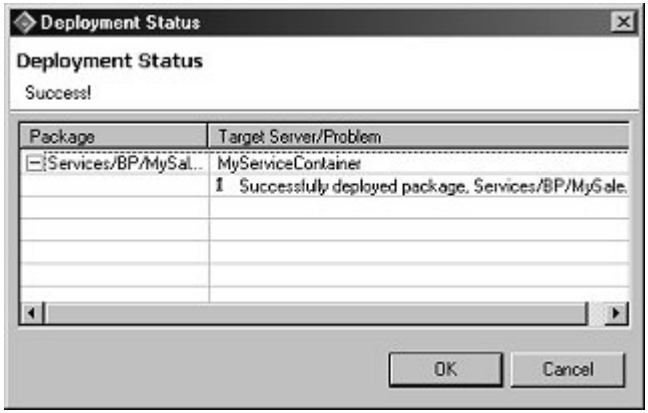
A command window appears. The UO51Runtime Adaptive Server Anywhere database starts and the database icon appears in the Windows system tray. Minimize any command windows.

2 In the **WorkSpace Navigator**, expand **MySybStore_Tutorials/Services/BP**, right-click **MySalesBPService.svcpkgdef,** and select **Deploy Package** from the context menu. In the **Select Target Server** dialog box, choose **MyServiceContainer** and click **OK**.

Note If a message prompts whether you want to overwrite an existing package file, click **Yes to All**.

The Console shows the progress of the operation.

3 When a message states that the deployment was successful, click **OK**.



4 Close the **Console** view by clicking the "**X**" on the windows title tab.

Lesson 3: Testing the service

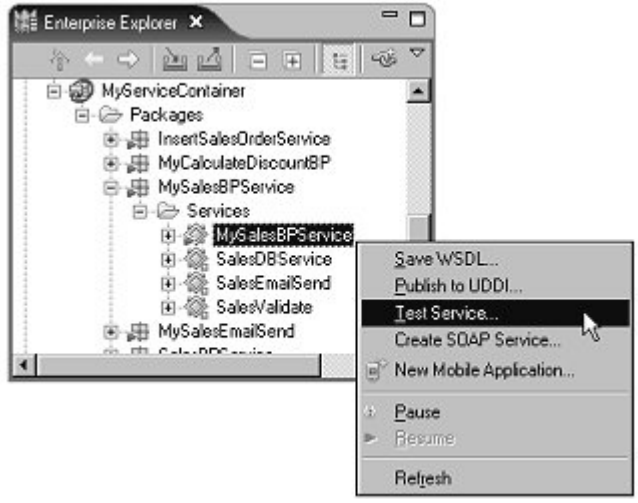
This lesson teaches you how to test the business process service you deployed in the previous lesson.

1 In the **Enterprise Explorer**, expand **Service Containers**, right-click **MyServiceContainer** and select **Connect** to establish a connection to the Unwired Orchestrator in WorkSpace.

   When the connection is established, you see a Packages folder beneath the MyServiceContainer connection profile.

2 In the **Enterprise Explorer**, expand **Service Containers/MyServiceContainer/Packages/MySalesBPService/Services.**

3 Right-click **MySalesBPService** and select **Test Service** from the context menu.



4 Select **Dialog Service Test Wizard** in the **Service Testing Wizard** window and click **Next**.

5 In the **Options** window, accept the defaults and click **Next**.

6 In the **Select a Method to Test** window, select **SalesDetailResponse manageInventory** and click **Next**.

7 In the **Parameters for Method** window, click **Edit** in the **Input Parameters** section.

8 In the **Specify Values of the Complex Type** dialog box, enter these values:

   - ItemNum – `A6459`
   - Qty – `4`

- Price – `199.99`
- SalesID – `13`

Click **OK**.

9 Click **Invoke** in the **Parameters for Method** window. You see this message in the Output Parameters pane:

```
<SalesDetailResponse>
<processingResult>SUCCESS</processingResult>
<failureReason>Successful execution</failureReason>
</SalesDetailResponse>
```

10 Click **Finish**.

Upon successful completion, an e-mail message is sent to the user specified in the e-mail message parameters.

Debugging a business process service

This tutorial shows you how to add tracing to a business process service so that you can validate and correct its logic.

The Unwired Orchestrator server writes messages to its log file as services execute. You can enable diagnostic tracing for a business process service by creating a Java service that writes messages to the Unwired Orchestrator log file, and invoking that Java service from the business process service.

This tutorial teaches you how to modify a business process service to write additional information to the log file about what it is doing while it executes.

This tutorial consists of:

Lesson 1: Creating a Java service to write tracing to the log
Lesson 2: Adding tracing to a business process service

Lesson 1: Creating a Java service to write tracing to the log

In this lesson, you create a Java service that writes trace information to the Unwired Orchestrator log file. Then you use the Java service in the business process service you created in previous tutorials to trace the logic.

1 Open the **Service Development** perspective.

2 Select **File|New|Service** from the WorkSpace main menu bar.

3 When the **Create a Service** wizard appears, select **Java Service** and click **Next**.

4 Select and expand **MySybStore_Tutorials/Services/Java** to populate the **Enter or Select the Parent Folder** field.

5 Enter `MyDebugJavaService` in the **File Name** field and click **Next**.

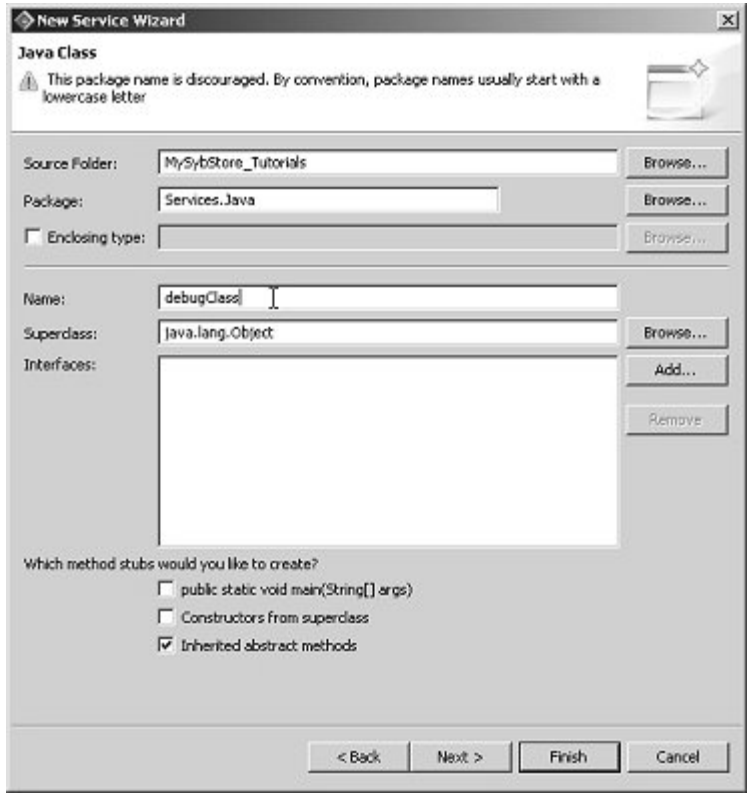
6 When the **Service Summary** window opens, click **Next**.

7 In the **Implementation Type** window, select **From New Source File (.java)** and click **Next**.

8 In the **Java Class** window, enter these values:

- Source folder – `MySybStore_Tutorials`
- Package – `Services.Java`
- Name – `debugClass`

- Superclass – `java.lang.Object`

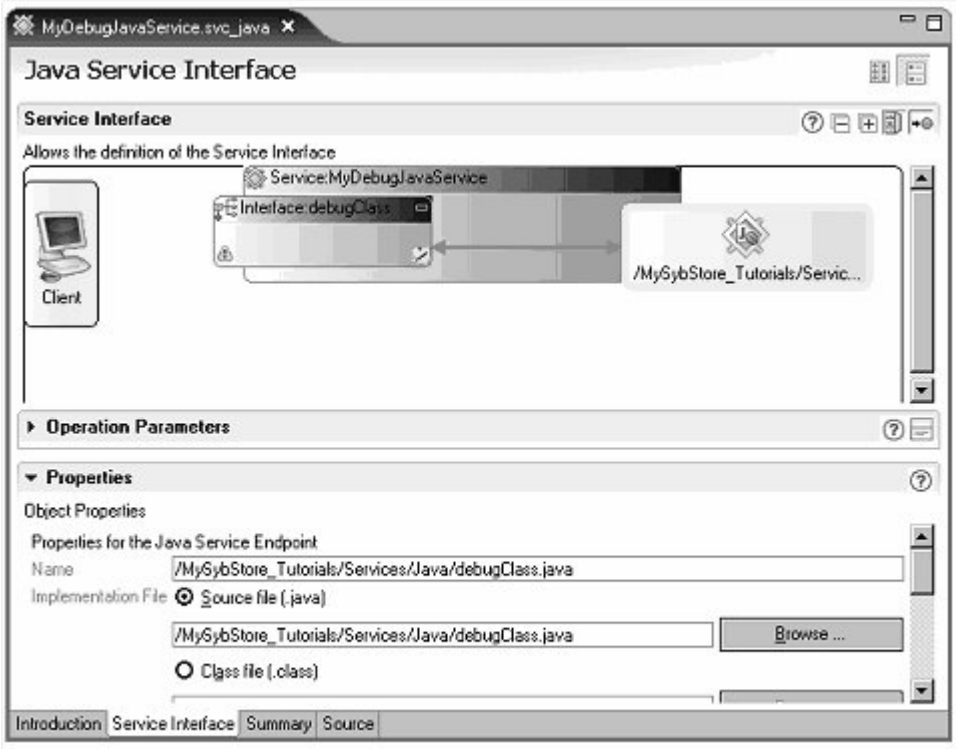


9 Select the **Inherited Abstract Methods** option if it is not selected.

10 Click **Finish**.

The new service is created and opens in the Java Service Editor.

11 Select the **Service Interface** tab in the service editor.



12 Create an operation:

   a In the **Java Service Interface** diagram, right-click the **Interface:debugClass** operation box and select **Add Operation|Create a New Method** from the context menu.

   b Select the new **operation1** method in the diagram, then in the **Properties** pane, enter `printMyInt` in the **Name** field.

13 Add two parameters:

   a Select the new **printMyInt** method in the diagram, and click **Add** in the **Operation Parameters** section to add a new parameter to the operation.

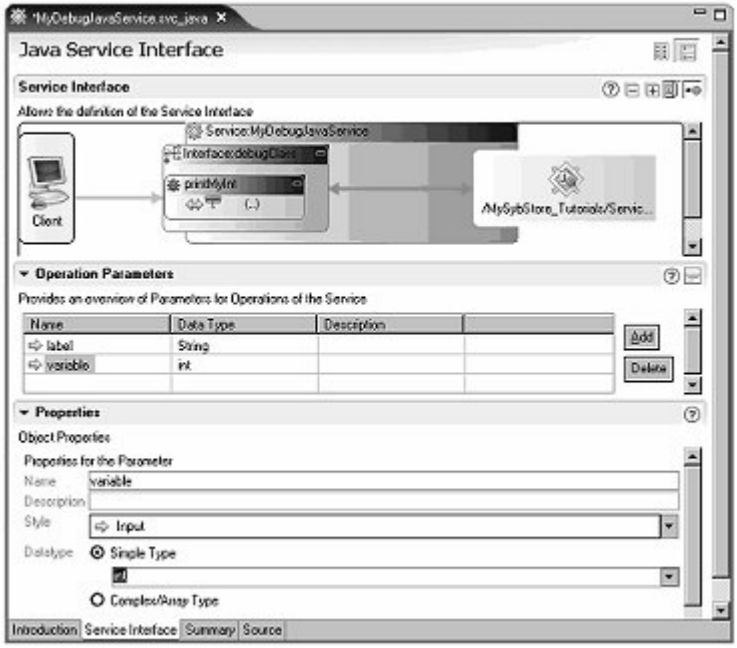
   b In **Operation Parameters** table, enter `label` in the **Name** field.

   c In the **Properties** pane, select `Input` for the **Style**, select **Simple Type** as the **Datatype** and select String from the **Data Type** drop-down list.

d Select the new **printMyInt** method again in the **Java Service Interface** diagram and click **Add** in the **Operation Parameters** section to add a second parameter to the operation.

e In **Operation Parameters**, enter `variable` in the **Name** field.

f In the **Properties** pane (below **Operation Parameters**), select `Input` for the **Style**, select **Simple Type** as the **Datatype** and select **int** from the **Datatype** drop-down list.



14 Repeat steps 12 and 13 to create a second operation called `printMyString` to write a label and the value of a string variable to standard output. Add two parameters with these values:

Parameter 1:

- Name – `label`
- Datatype – **Simple Type** String

Parameter 2:

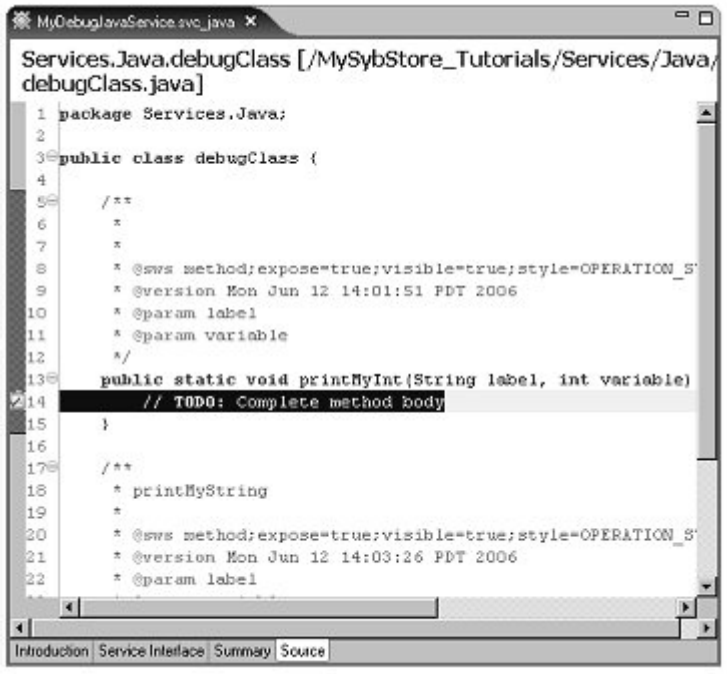
- Name – `variable`

- Datatype – **Simple Type** String

15 Modify the source code of the PrintInt method to carry out the service's operation:

   a Select the **Source** tab in the Java Service Editor.

b In the code window, select the line below `public static void printMyInt` that says "`// TODO: Complete method body`".

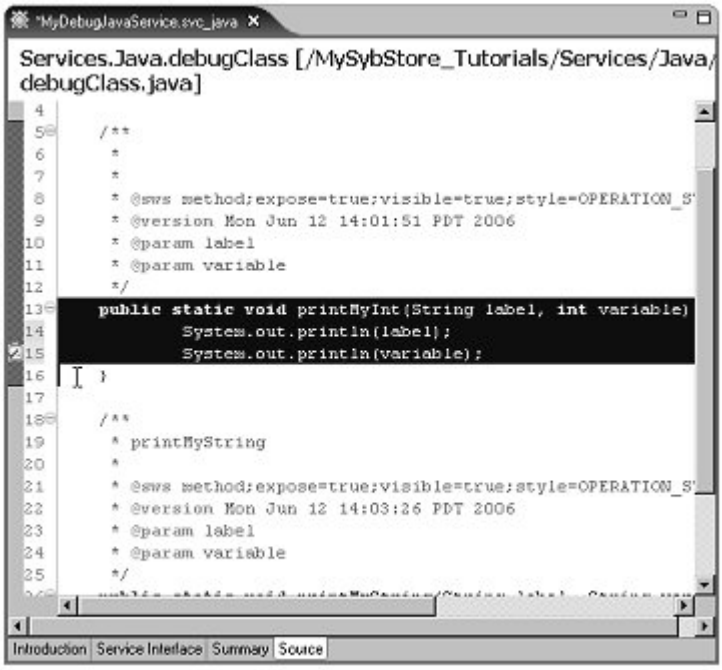


c Replace this line with the following code:

```
System.out.println(label);
System.out.println(variable);
```

The method now reads as follows. See the following graphic for reference.

```
public static void printMyInt(String label, int variable) {
                    System.out.println(label);
                    System.out.println(variable);
```



16 Select **File|Save** from the WorkSpace main menu bar.

17 Build and deploy the package for MyDebugJavaService._svc_java.

   a In the **WorkSpace Navigator**, expand **MySybStore_Tutorials/Services/Java**, right-click **MyDebugJavaService.svc_java**, and select **Create Sybase Services Package Profile** from the context menu.

      The new package is created and opens in the Sybase Services Package Profile Editor.

b In the **WorkSpace Navigator**, right-click **MyDebugJavaService.svcpkgdef** and select **Build Package** from the context menu. This builds the actual package from the package profile you created earlier in the lesson.

Note If a message prompts whether you want to overwrite an existing package file, click **Yes to All**.

The Console view opens and shows the progress of the operation.

c When a message states that the package was successfully created, click **OK**.

d If Unwired Orchestrator is not running, use the Windows taskbar and select **Start|Programs|Sybase|Sybase WorkSpace|UO 5.1|Start UO**.

e In the **WorkSpace Navigator**, expand **MySybStore_Tutorials/Services/Java**, right-click **MyDebugJavaService.svcpkgdef,** and select **Deploy Package** from the context menu. In the **Select Target Server** dialog box, choose **MyServiceContainer** and click **OK**.

Note If a message prompts whether you want to overwrite an existing package file, click **Yes to All**.

The Console shows the progress of the operation.

f When a message states that the deployment was successful, click **OK**.

g Close the **Console** view by clicking the "**X**" on the windows title tab.

18 Select **File|Close** to close the editor.

When this service runs in the context of the Unwired Orchestrator server, the data is written to the Unwired Orchestrator server log file. By default, this log file is found at *%WS_INSTALL_DIR%\DevRuntimes\EAServer\bin\Jaguar.log*.

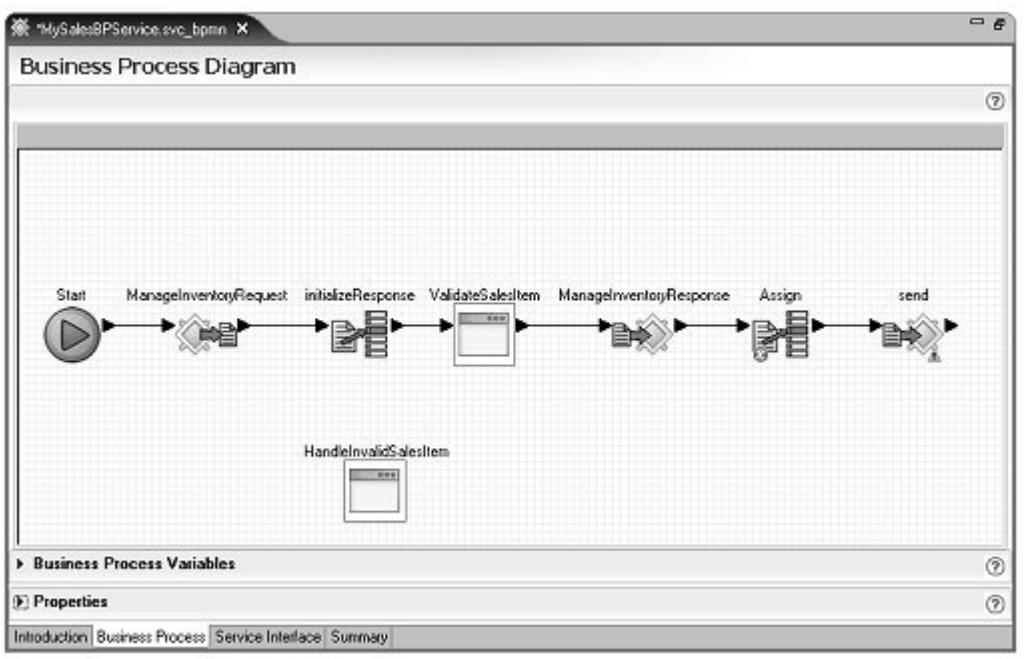
Lesson 2: Adding tracing to a business process service

In this lesson, you add an invocation of printMyInt to SalesBPService to log the value of a variable returned from another service invocation.

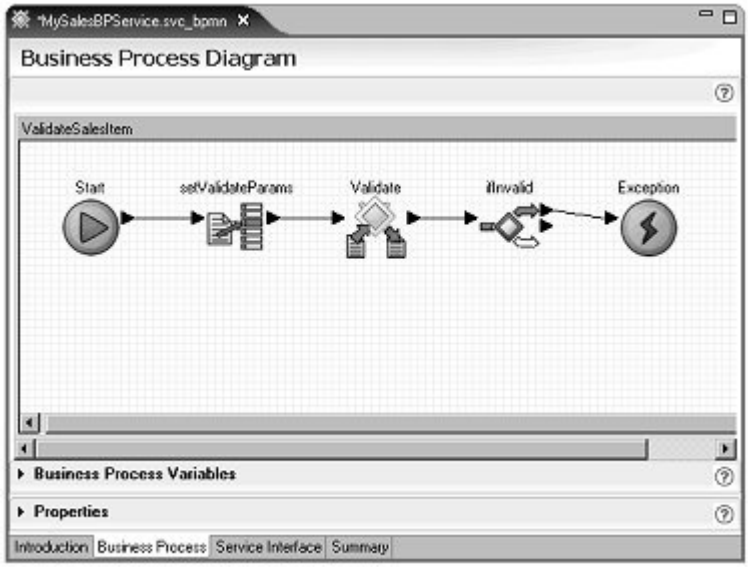
In the preceding lesson you created a Java service with two operations:

- printMyInt writes the value of an int type variable and a label to the Unwired Orchestrator server log file.
- printMyString writes the value of a string type variable and a label to the Unwired Orchestrator server log file.

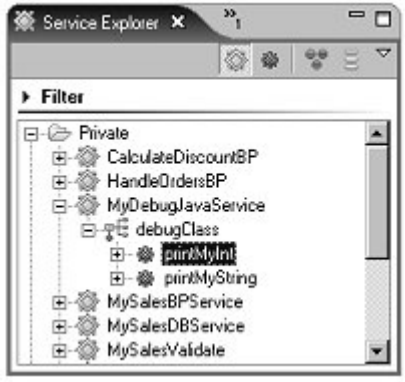
1 In the **WorkSpace Navigator**, expand **MySybStore_Tutorials/Services/BP** and double-click **MySalesBPService.svc_bpmn** to open the service in the editor.



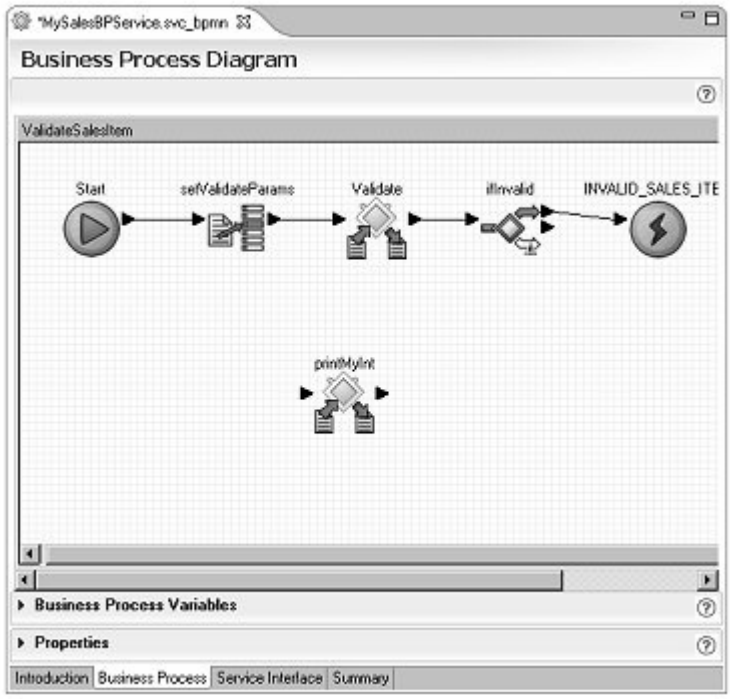
2 Double-click the **ValidateSalesItem** activity, then maximize that operation's canvas.



3 Open the **Service Explorer**, and expand **Private/MyDebugJavaService/debugClass** and select **printMyInt**.

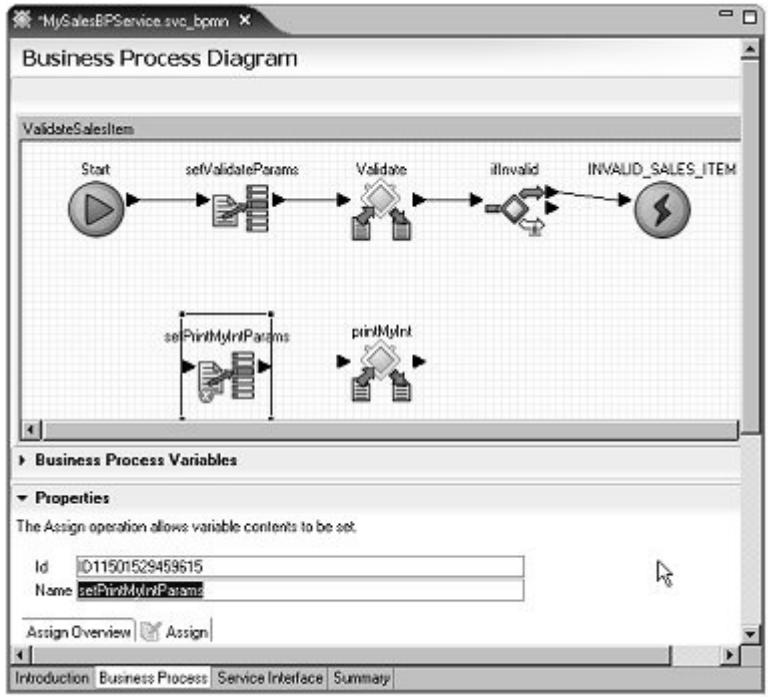


4 Drag **printMyInt** onto the business process canvas.



5 Right-click in the canvas and select **Show Tool Palette**, select the **Activities** category, then drag an **Assign** activity from the **Tool Palette** onto the canvas.

6 Select the **Assign** activity on the editor canvas, expand the **Properties** pane, and rename the **Assign** activity to `setPrintMyIntParams`.



7 In the **Properties** pane, click **New** to add a second **Assign** variable in the **Assign Overview** table.

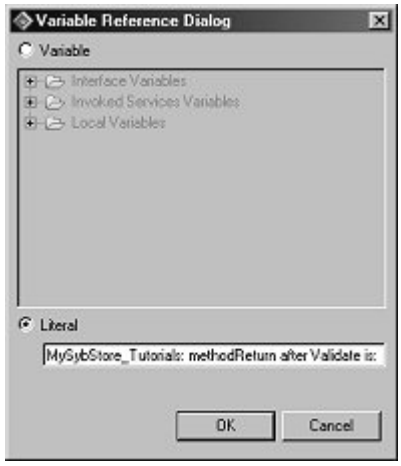
8 Set the Source and Target properties of the first Assign variable:

a Click the ellipsis button in the **Source** column of the first **Assign** variable.

b In the **Variable Reference Dialog**, select the **Literal** option and enter:

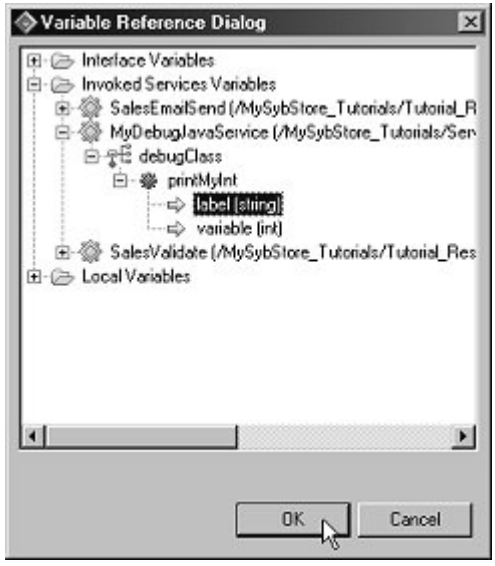
```
MySybStore_Tutorials: validateReturn after Validate is:
```

Click **OK**.



c Click the ellipsis button in the **Target** column of the first **Assign** variable.

d In the **Variables Reference Dialog** box, select **Invoked Services Variables/MyDebugJavaService/debugClass/printMyInt/label** and click **OK**.



9 Set the Source and Target properties of the second Assign variable:

   a Click the ellipsis button in the **Source** column of the second **Assign**.

   b In the **Variables Reference Dialog** window, select **Invoked Services Variables/SalesValidate/SalesItem/Validate/ValidateReturn** and click **OK**.

   c Click the ellipsis button in the **Target** column of the second **Assign** variable.

   d In the **Variables Reference Dialog** box, select **Invoked Services Variables/MyDebugJavaService/debugClass/printMyInt/variable** and click **OK**.

10 Connect the setPrintMyIntParams activity and printMyInt service invocation into the activity flow. Rearrange the object as necessary to achieve the look of the graphic that follows.

   a Move **setprintMyIntParams** to the left of the **printMyInt** object in the editor canvas.

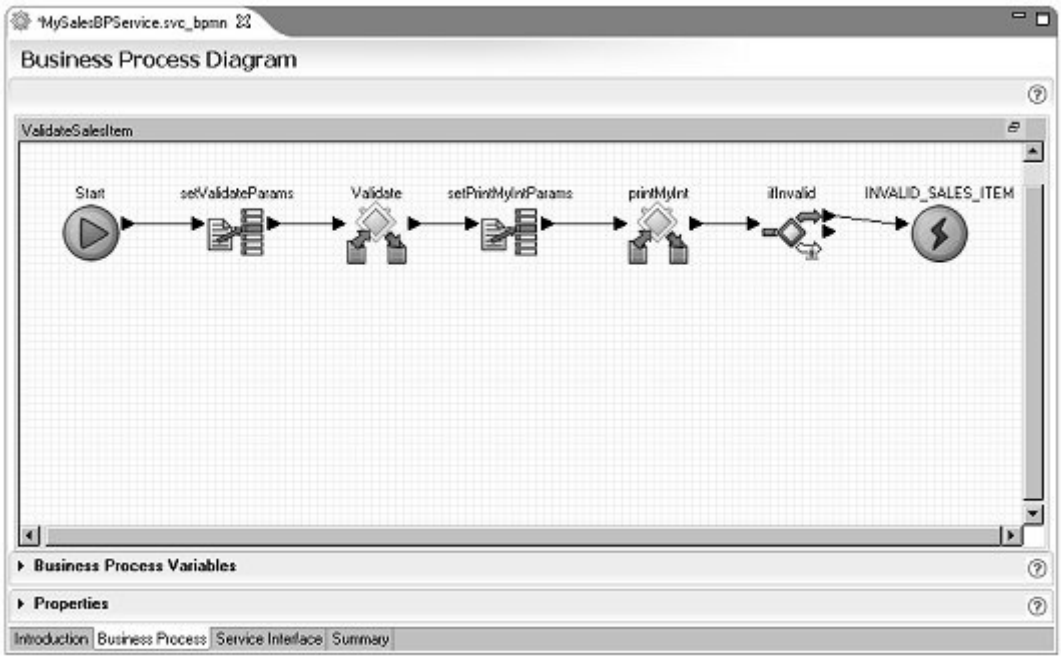
   b Click the right arrow on **setprintMyIntParams** and drag to and click the left arrow of **printMyInt** to connect those objects.

   c In the diagram, right-click the line between **Validate** and **ifInvalid** and select **Delete**.

   d Drag your cursor around **setprintMyIntParams** and **printMyInt** on the canvas to select those objects and their connection and move them between **Validate** and **ifInvalid**.

   e Click the right arrow of **Validate** and drag to and click the left arrow of **setprintMyIntParams** to connect those objects.

f Click the right arrow of **printMyInt** and drag to and click the left arrow of **ifInvalid** to connect those objects.



11 Select **File|Save** from the WorkSpace main menu, then select **File|Close All**.

12 Use “Packaging, deploying, and testing a business process service” on page 143 to repackage and deploy **MySybStore_Tutorials/Services/BP/MySalesBPService.svcpkgdef**.

Note See Chapter 4, “Unwired Orchestrator Logging Tutorials,” for additional logging tutorials.

Creating a business process service correlation set

This tutorial demonstrates how to use a correlation mechanism in a business process service.

When a business process service invokes a one-way operation of another business process service, the originating service typically needs to wait for the response to come back from the called service. When a response is expected separately, the business process service must correlate the response back from the called service to the original request that was sent. To enable this activity, you create a correlation set.

This tutorial consists of:

Lesson 1: Creating a business service to send a correlation request
Lesson 2: Creating a business process service correlation set
Lesson 3: Adding order-processing logic
Lesson 4: Adding logging activities
Lesson 5: Sending a correlated response to the initiating business process

Lesson 1: Creating a business service to send a correlation request

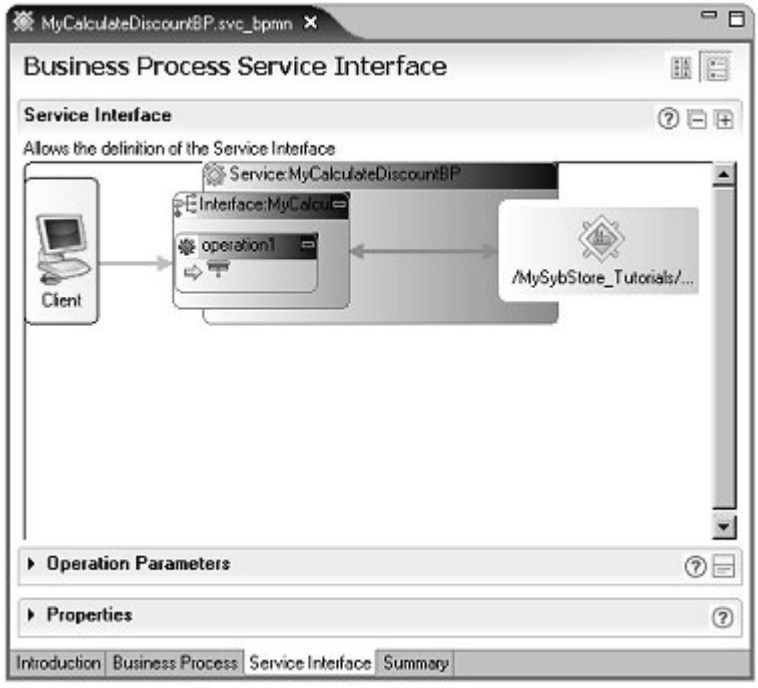
1 Select **Window|Open Perspective|Other**, select **Service Development**, and click **OK**

2 Select **File|New|Service** from the WorkSpace main menu bar.

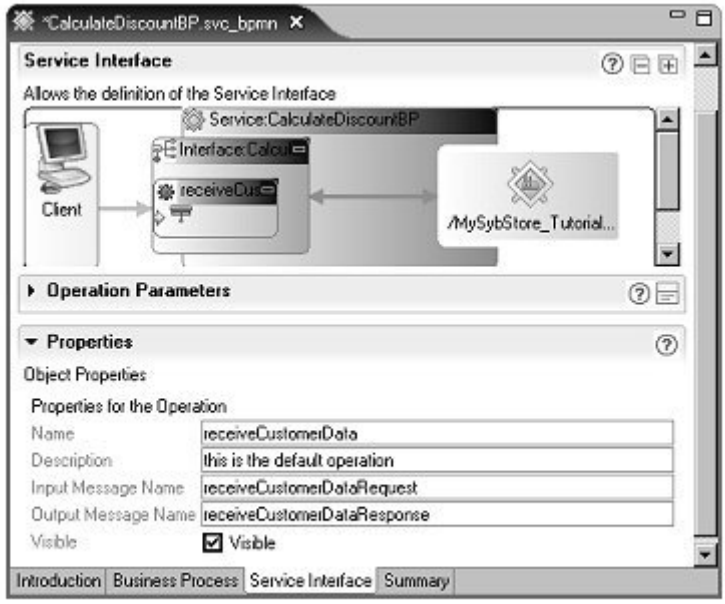
3 When the **Create a service** wizard opens, select **Business Process Service** and click **Next**.

4 Select the **MySybStore_Tutorials/Services/BP** as the parent folder, enter `MyCalculateDiscountBP` as the name of the service, and click **Finish**.

5 When the new service opens in the Business Process Service Editor, select the **Service Interface** tab.



6 Expand the **Properties** pane, select the **operation1** box in the **Service Interface** diagram, and change the operation **Name** to `receiveCustomerData`.



7 Open the **Operation Parameters** pane, reselect the **receiveCustomerData** operation in the diagram if necessary, and click **Add** three times in the **Operation Parameters** pane to add three parameters with these values:

Parameter 1:

- Name – `customerid`
- Style – Input
- Datatype – **Simple Type** string

Parameter 2:

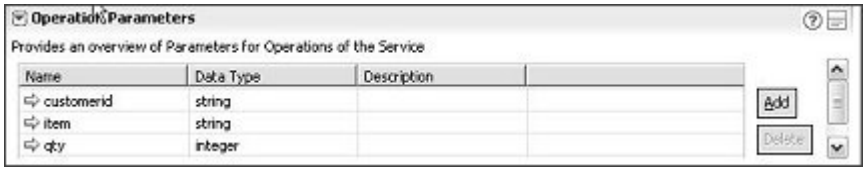
- Name – `item`
- Style – Input
- Datatype – **Simple Type** string

Parameter 3:

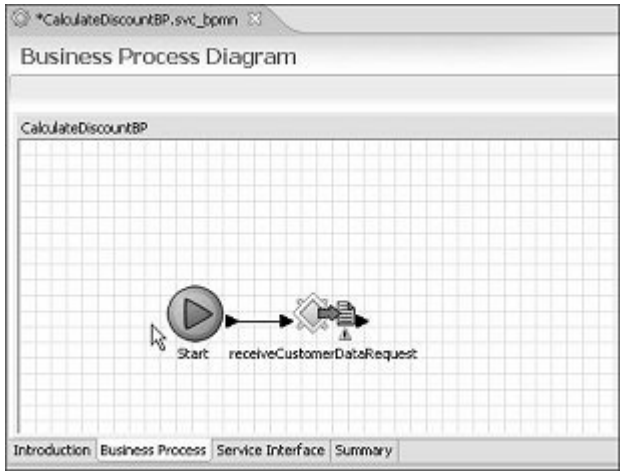
- Name – `qty`

- Style – Input
- Datatype – **Simple Type** integer

Note To change parameter name and type, use the **Properties** pane.



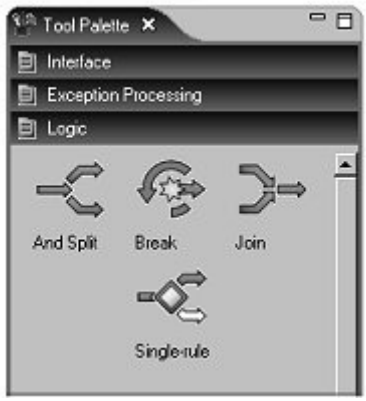
8 Select the **Business Process** tab.



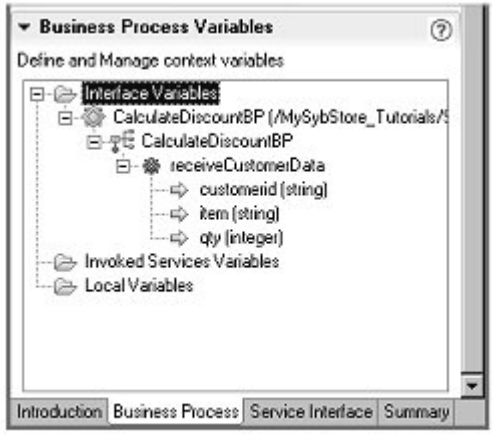
9 Add some logic to calculate the discount. Right-click in the editor canvas and select **Show Tool Palette** from the context menu.

Note If the Tool Palette disappears, click the Tool Palette icon in the Fast View to redisplay it.

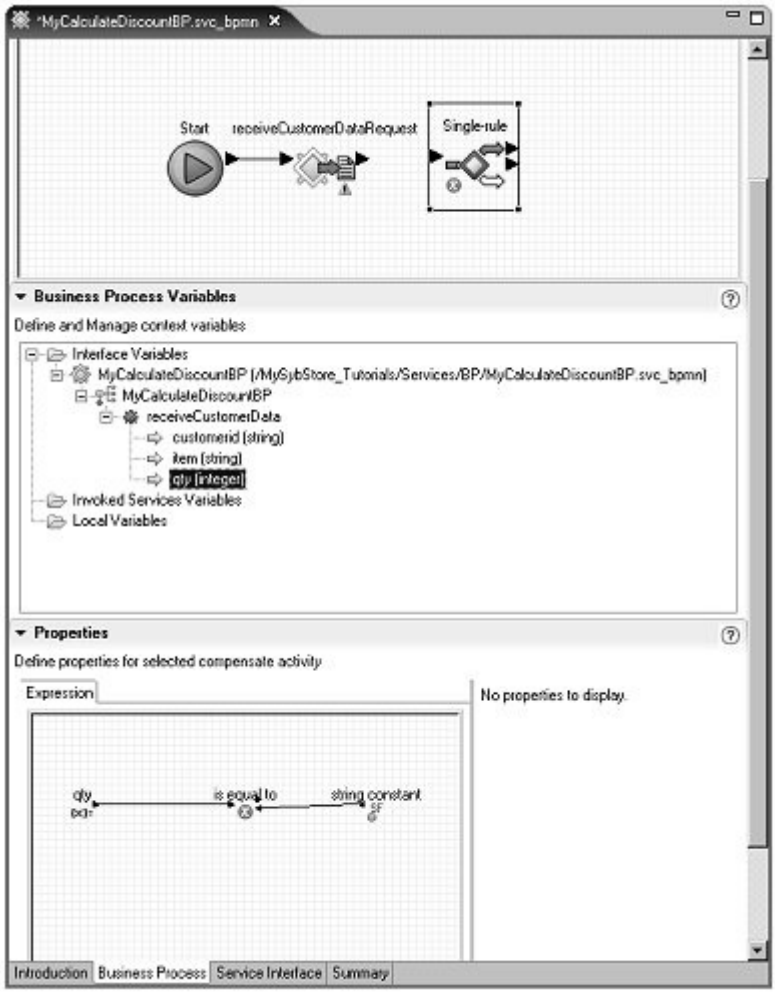
10 Select the **Logic** category and drag and drop the **Single-rule** logic onto the canvas.



11 Select the **Single-rule** object in the diagram and expand the **Business Process Variables** pane in the editor. Expand the **Interface Variables** folder, then expand the tree view below that until you see the **qty** parameter.



12 With the **Single-rule** object selected in the diagram, expand the **Properties** pane and drag the **qty** variable to the **Expression** editor in the **Properties** pane and drop it on the **Select Variable** icon.



13 Click in the **Expression** editor, reselect or reopen the **Tool Palette**, select the **Boolean Functions** category, and drag and drop the **is greater than** item on top of the **is equal to** item in the **Expression** editor to replace it.

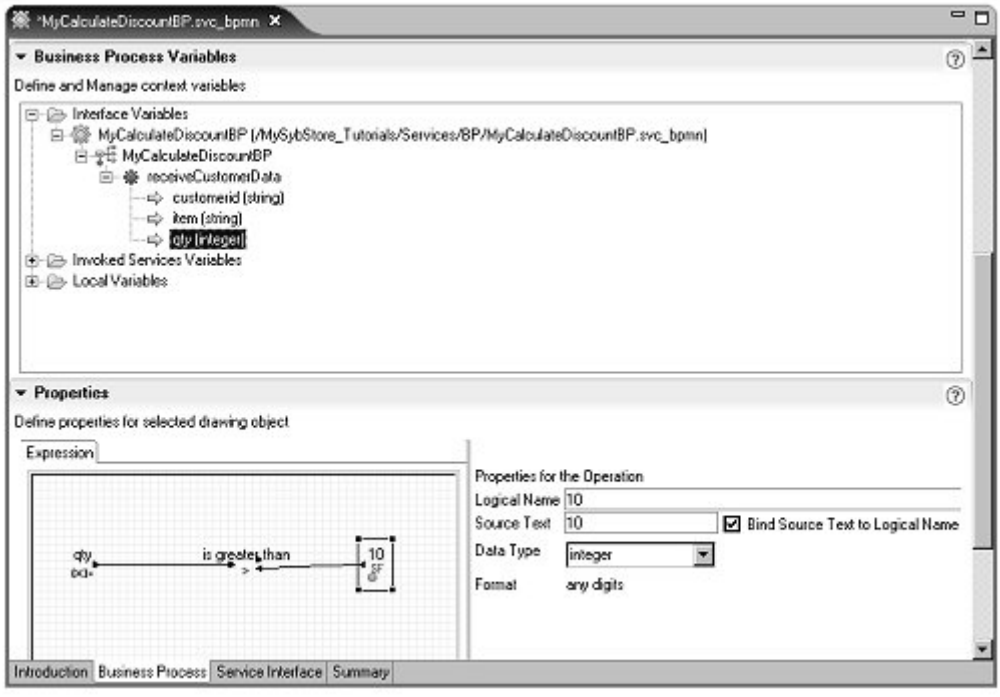
Note To redisplay the Tool Palette from the Fast View, right-click the Tool Palette title tab and select Fast View. You can also right-click the Tool Palette title tab, select Detached and move the Tool Palette view anywhere on your screen.

---

14 Click in the **Expression** editor, reselect or reopen the **Tool Palette**, select the **Generic Variable and Constants** category, and drag and drop the **integer constant** item on top of the **string constant** item in the Expression editor to replace it.

15 Select the **integer constant** item in the **Expression** editor, and in the **Properties for the Operation** section to the right of the **Expression** editor, enter `10` in the **Logical Name** field and select **Bind Source to Logical Name**. The **Source Text** field now displays "`10`."

The completed single-rule expression should look like this:



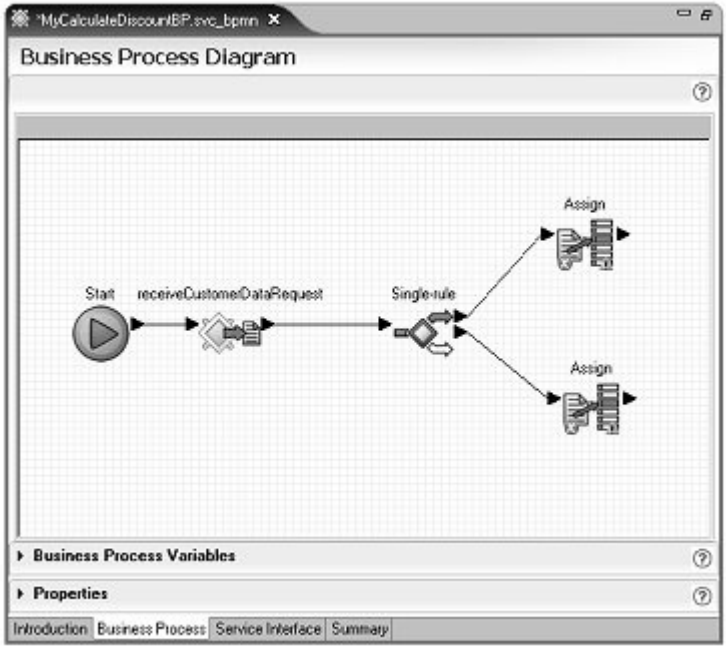
16 In the **Business Process Diagram** canvas, click the right arrow of **receiveCustomerDataRequest** and drag and click on the left arrow of **Single-rule** to connect the logic.

17 Close the **Properties** pane, then select **File|Save** on the WorkSpace main menu bar.

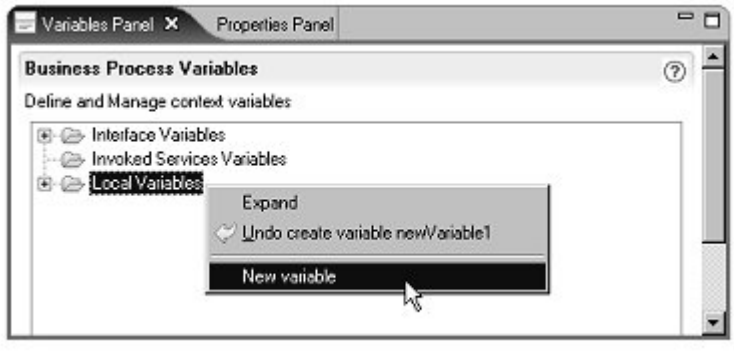
18 On the **Business Process Diagram** canvas, reopen the **Tool Palette**, select the **Activities** category, then drag and drop two **Assign** activities to the **MyCalculateDiscountBP** service.

19 Connect the two Assign activities to the Single-Rule activity. Rearrange the objects on the canvas as necessary to achieve the look shown in the following graphic.

- Click the right top arrow of **Single-rule** and drag and click on the left arrow of the first **Assign** to connect those objects.
- Click the right bottom arrow of **Single-rule** and drag and click on the left arrow of the second **Assign** to connect those objects.



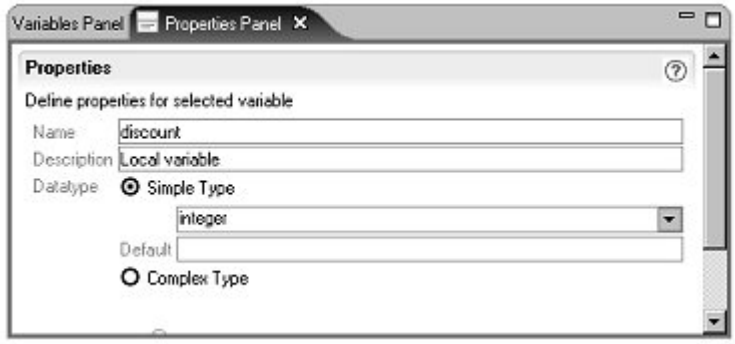
20 Expand the **Business Process Variables** pane (below the canvas), right-click the **Local Variables** folder and select **New Variable** from the context menu.



21 In the **Business Process Variables** pane, right-click the **Local Variables** folder and select **New Variable** from the context menu.

22 Expand the **Local Variables** folder, right-click the new local variable, and select **Edit Variable** from the context menu.

23 When the **Properties** section opens, change the **Name** to `discount` and verify that the **Datatype** is **Simple Type** Integer.



24 Select **File|Save** from the WorkSpace main menu bar.

25 Make the new discount variable the target for each Assign activity:

a On the **Business Process Diagram** canvas, select the top **Assign** object, expand the **Properties** pane, then expand the **Business Process Variables** pane.

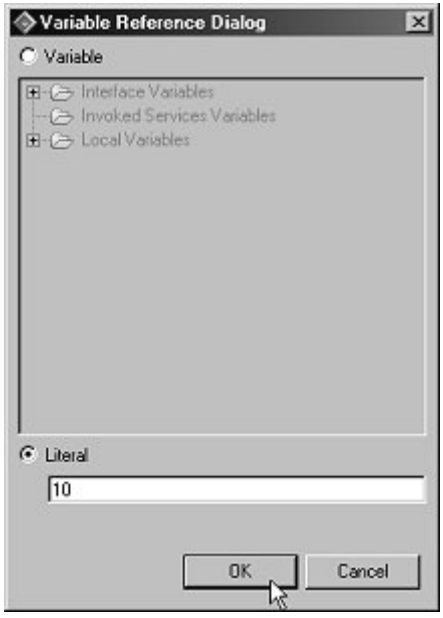
b Drag **Local Variables/discount** from the **Business Process Variables** pane and drop it in the **Target** column of the top **Assign**.

c Select the bottom Assign object on the canvas, and drag **Local Variablesdiscount** from the **Business Process Variables** pane and drop it in the **Target** column of the bottom **Assign**

26 Give the top Assign activity discount a literal value of 10.

a Select the top **Assign** on the canvas and in the **Properties** pane, click the ellipsis button in the **Source** column.

b When the **Variable Reference Dialog** window opens, select the **Literal** option, enter `10`, and click **OK**.



27 Set the bottom Assign activity discount variable a literal value of 0 (zero).

a Select the bottom **Assign** on the canvas and in the **Properties** pane, click the ellipsis button in the **Source** column.

b When the **Variable Reference Dialog** window opens, select the **Literal** option, enter `0` (zero), and click **OK**.

28 Select **File|Save** from the WorkSpace main menu bar, then select **File|Close** to close the editor.

29 In the **WorkSpace Navigator**, create the **MyCalculateDiscountBP** package profile and package, then deploy the service.

Note Unwired Orchestrator must be running and there must be a connection established from MyServiceContainer in WorkSpace. See "Starting and connecting to the Unwired Orchestrator server" on page 19 for instructions.

a Expand **MySybStore_Tutorials/Services/BP**, right-click **MyCalculateDiscountBP.svc_bpmn** and select **Create Sybase Services Package Profile** from the context menu. Select **File|Close** to close the package profile in the editor.

b Right-click **MyCalculateDiscountBP.svcpkgdef** and select **Build Package** from the context menu. When a message displays stating that the package was built successfully, click **OK**.

c Right-click **MyCalculateDiscountBP.svcpkgdef** and select **Deploy Package** from the context menu. When the **Select Target Server** window opens, select **MyServiceContainer** and click **OK**. If a message asks if the current package file can be overwritten, click **Yes**.

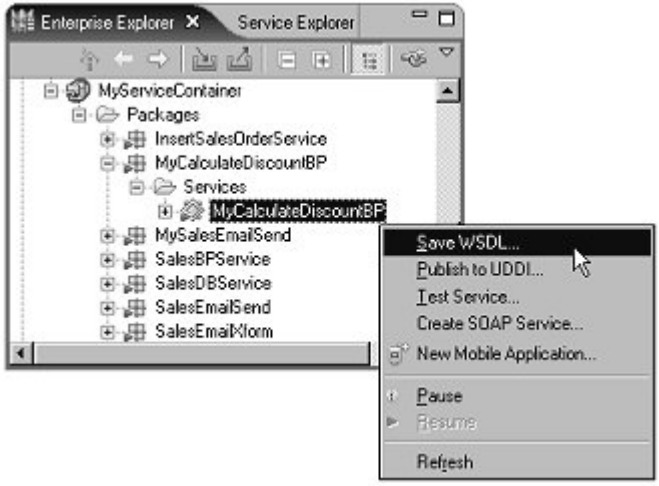
d When the **Deployment Status** window states that deployment to MyServiceContainer was successful, click **OK**.

The Console view shows the progress of each activity. Click the "**X**" on the Console title tab to close that window.

Note See the WorkSpace online help topic *WorkSpace Development|Service Development|Packages* for more information about building and deploying service packages.

30 In the **Enterprise Explorer**, expand **Service Containers/MyServiceContainer/Packages/MyCalculateDiscountBP/Services** and locate **MyCalculateDiscountBP**.

31 Right-click **MyCalculateDiscountBP** and select **Save WSDL** from the context menu.



32 When the **Save WSDL Wizard** dialog box opens, select **MySybStore_Tutorials/Services/BP** in the tree view, confirm the **File Name** is `MyCalculateDiscountBP`, and click **Finish**.

33 In the **WorkSpace Navigator**, expand **MySybStore_Tutorials/Services/BP** right-click **MyCalculateDiscount.wsdl**, and select **Create SOAP Service** from the context menu.

34 When the **New Service Wizard** opens, select **MySybStore_Tutorials/Services/BP** as the parent folder, enter `MyDiscountSOAP` for the **File Name** of the service, and click **Next**.

35 In the **Service Summary** window, click **Next**.

36 In the **Service Endpoint Creation** window, select the option **Yes, Create An Endpoint Now**, and click **Next**.

37 In the **Endpoint Name** window, accept the default **Name** `endpoint` and click **Next**.

38 In the **Connection Properties** window, click **Finish**.

39 The SOAP service is created and opens in the SOAP Service Editor. Select **File|Close**. You do not need to save the service because you have not made any changes to it.

Lesson 2: Creating a business process service correlation set

1. Select **Window|Open Perspective|Other** from the WorkSpace main menu, select **Service Development** in the **Select Perspective** window, and click **OK**.
2. Select **File|New|Service** from the WorkSpace main menu bar.
3. When the **Create a Service** wizard opens, select **Business Process Service,** and click **Next**.
4. Select the **MySybStore_Tutorials/Services/BP** as the parent folder, enter `MyHandleOrdersBP` for the service name, and click **Finish**.
5. When the business process service opens in the editor, select the **Service Interface** tab.
6. Select **operation1** box in the diagram, then expand the **Properties** pane and change the **Name** to `process`.
7. Open the **Operation Parameters** pane and click **Add** three times to add three parameters with these values:

   Parameter 1:

   - Name – `customerid`
   - Style – Input
   - Datatype – **Simple Type** string

   Parameter 2:

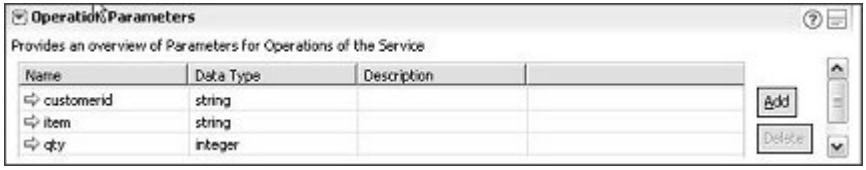
   - Name – `item`
   - Style – Input
   - Datatype – **Simple Type** string

   Parameter 3:

   - Name – `qty`
   - Style – Input

- Datatype – **Simple Type** integer

Note To change parameter name and type, use the **Properties** pane.



8 Select **File|Save** from the WorkSpace main menu.

Lesson 3: Adding order-processing logic

In this lesson, you add order-processing logic that calls the discount service that you created in lesson 1 of this tutorial.

The business process waits for the discount data to come back via a correlated response. Upon receiving the discount data, the order processing completes.

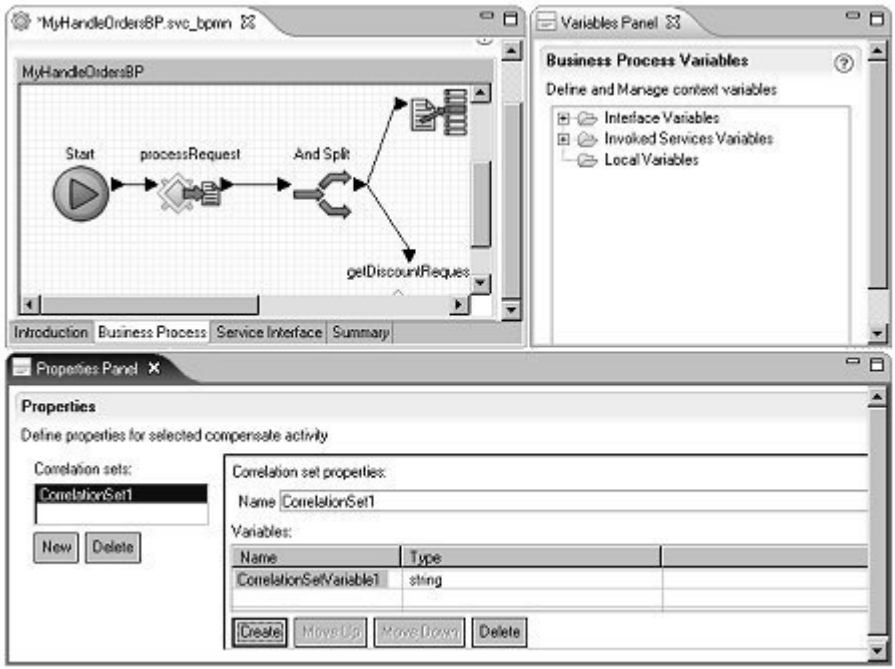
1 In the **WorkSpace Innovator**, expand **MySybStore_Tutorials/Services/BP** and double-click **MyHandleOrdersBP.svc_bpmn** to open the file in the Business Process Service Editor.

2 In the Business Process Service Editor, select the **Business Process** tab.

3 Arrange the Properties and Variables panes so you can work with them more easily. Right-click in the Business Process Service Editor canvas and select **Show Properties Panel**, then right-click again and select **Show Variables Panel** to move the panels into outside views.

Note You can also move the views to other locations in the WorkSpace main window for ease of use.

4 Click anywhere on the editor canvas to have the **Properties Panel** display **Properties**, click **New** beneath the **Correlation Sets** pane, then click **Create** below the **Variables** table.



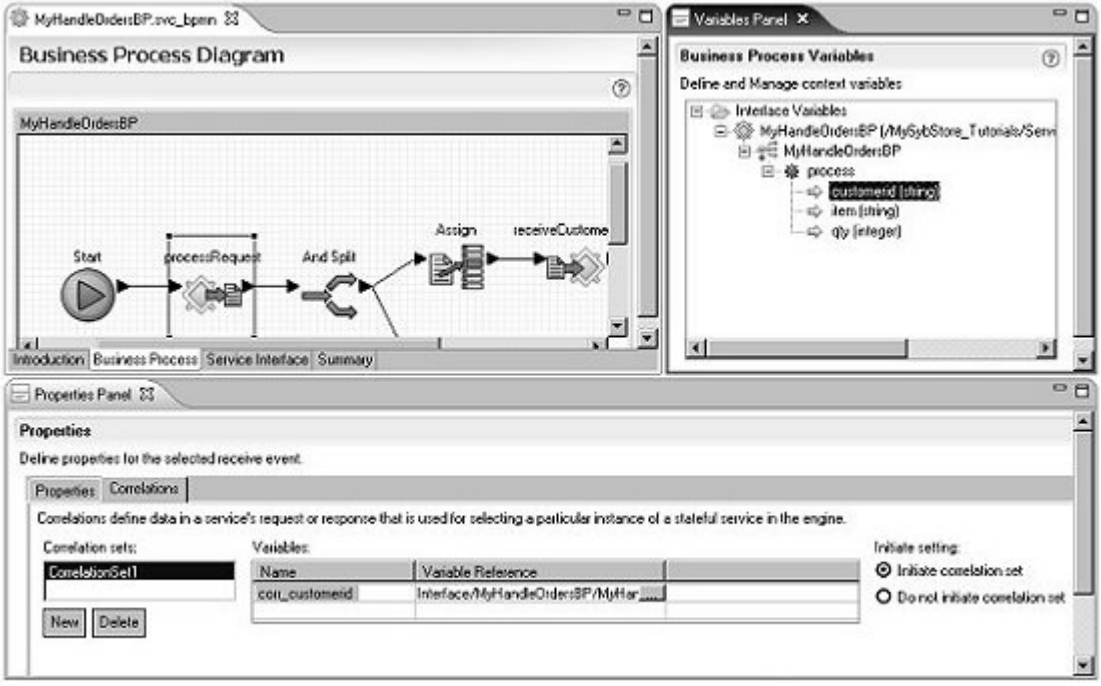
5 Enter `corr_customerid` in the variable **Name** column.

6 Select the **processRequest** activity in the editor canvas, expand the **Properties** pane (not the Properties Panel), and select the **Correlations** tab.

7 Click **New** below the **Correlation Sets** pane. By default, the correlation set you just created (CorrelationSet1) displays.

8 Expand the **Business Process Variables** view of the **Variables Panel** and drag and drop **customerid** to the **Variable Reference** column in the **Properties** pane.

9 Click **Initiate Correlation Set** on the far right of the **Variables** table. You may have to maximize the WorkSpace window to see this option.



10 Right-click in the editor canvas and select **Show Tool Palette**. In the following order, drag and drop these objects to the editor canvas.

(Tool Palette category | [Logic|Activity|Exception])

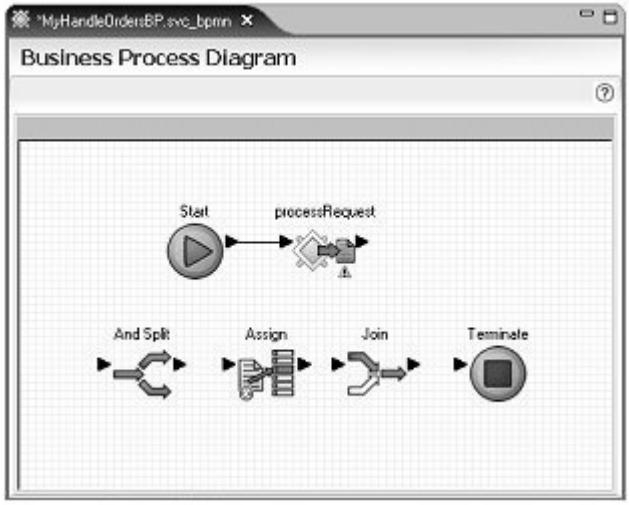
a **Logic** | **And Split**

b **Activities** | **Assign**

c **Logic** | **Join**

d **Exception Processing** | **Terminate**

The editor canvas should look like this:



11 In the **Service Explorer**, expand **Public/MyDiscountSOAP/MyCalculateDiscountBP** and drag **receiveCustomerData** to the editor canvas and drop it between the **Assign** and **Join** objects.

12 On the editor canvas, horizontally line up the **And Split**, **Assign**, **receiveCustomerData**, **Join**, and **Terminate** objects after the **processRequest** object.

13 Join the following business logic activities. Rearrange the objects on the canvas as necessary to achieve a look similar to the graphic that follows.

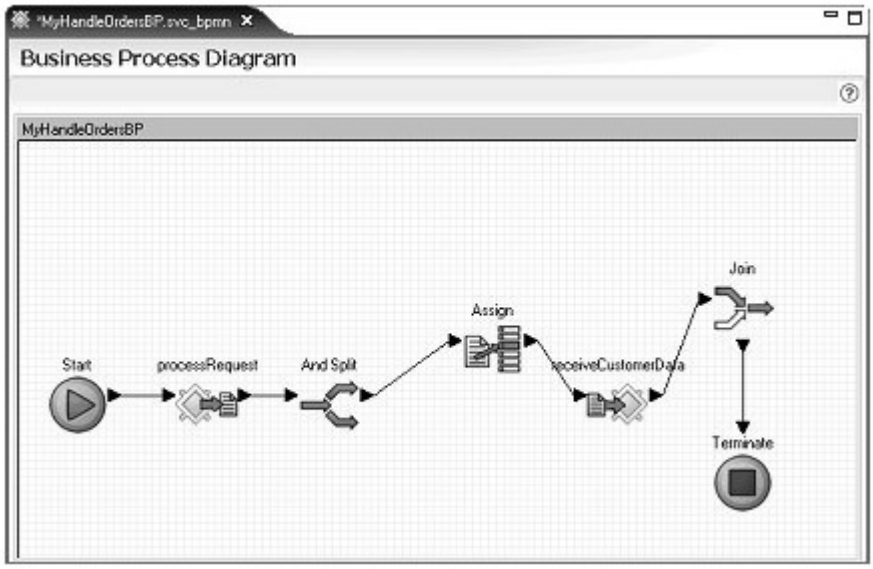
   a Click the right arrow of **processRequest**, drag to and click the left arrow of **And Split**.

   b Click the right arrow of **And Split**, drag to and click the left arrow of **Assign**.

   c Click the right arrow of **Assign**, drag to and click the left arrow of **receiveCustomerData**.

   d Click the right arrow of **receiveCustomerData**, drag to and click the left arrow of **Join**.

e Click the right arrow of **Join**, drag to and click the left arrow of **Terminate**.

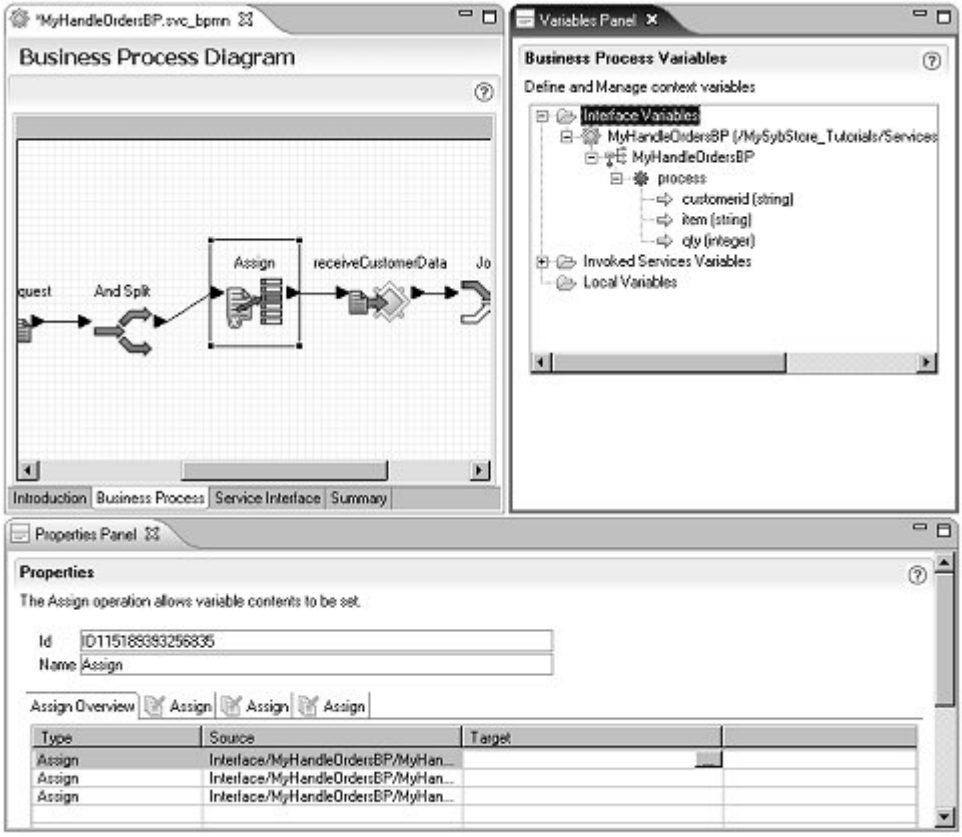


14 In the editor canvas, select the **Assign** object and expand the **Properties Panel**

15 In the **Properties Panel**, click **New** (below the **Assign Overview** table) twice to create two additional assignments.

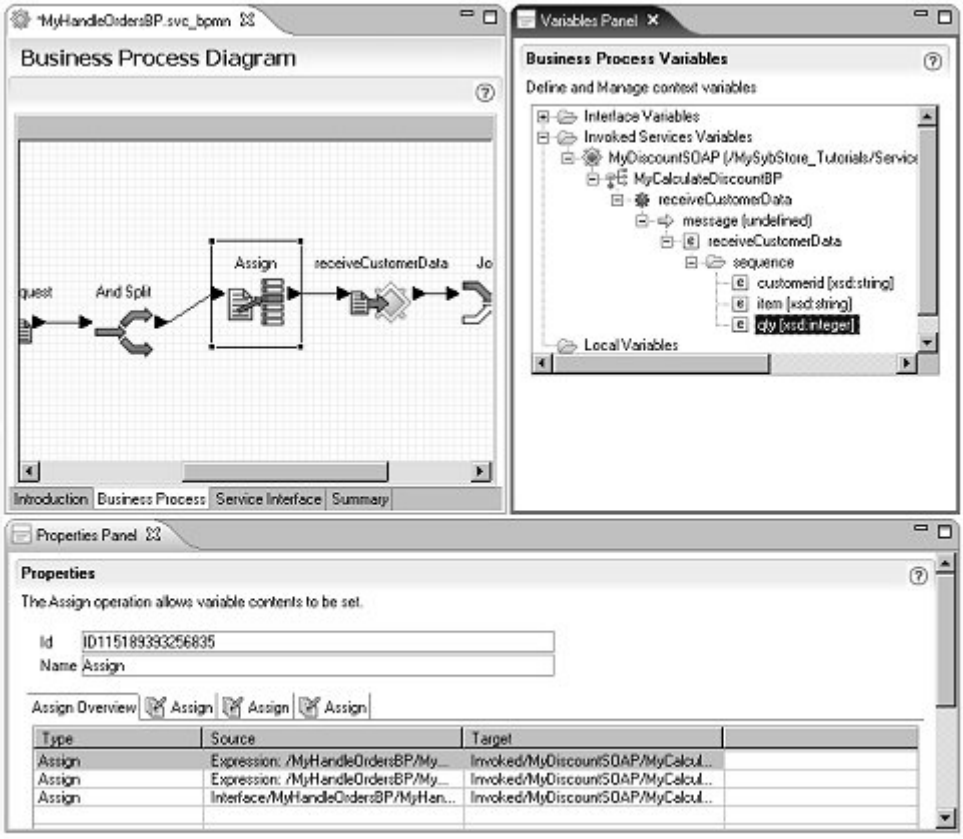
16 In the **Variables Panel**, and expand **Interface Variables/MyHandleOrdersBP/MyHandleOrdersBP/process** in the tree view.

17 Drag and drop the **customerid**, **item**, and **qty** parameters to the **Source** column of the three new assignments.



18 In the **Variables Panel** expand **Invoked Services Variables/MyDiscountSOAP/MyCalculateDiscountBP/receiveCustomerData/message/receiveCustomerData/sequence** in the tree view.

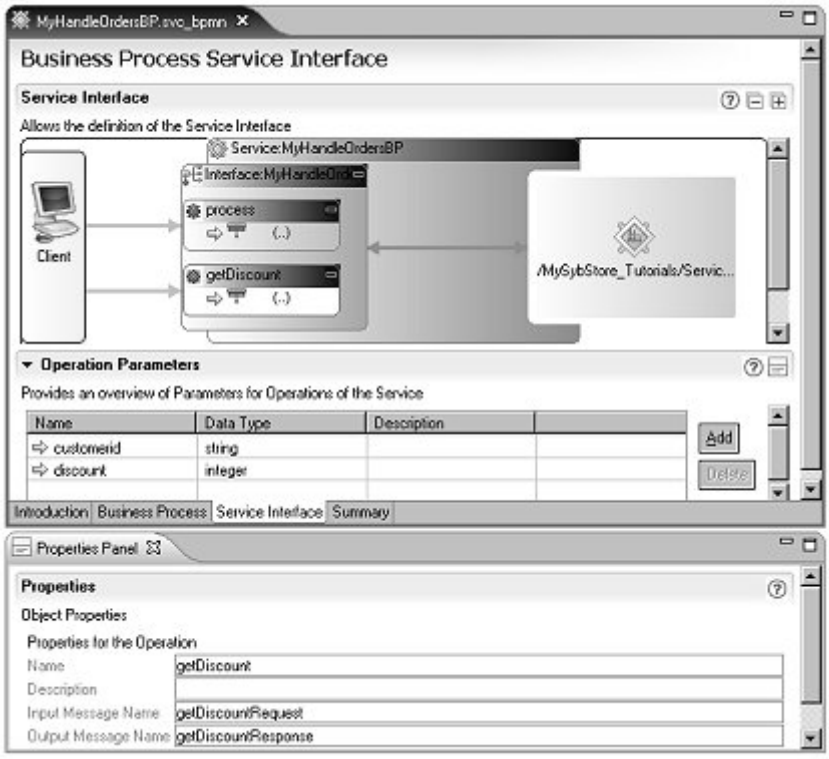
19 Drag and drop the **customerid**, **item**, and **qty** to the **Target** column of the three assignments.



20 Select **File|Save** from the WorkSpace main menu. When you select the editor canvas again, the red "**X**" that was on the Assign icon disappears because you have specified the Assign's target and source.

21 In the Business Process Service Editor, select the **Service Interface** tab, right-click in the **Interface:MyHandleOrdersBP** box in the diagram, and select **Add Operation** from the context menu.

22 Select the new operation, expand the **Properties Panel**, and enter `getDiscount` in the **Name** field.



23 Expand the **Operation Parameters** pane in the editor and select the new **getDiscount** operation in the diagram.

24 Click **Add** twice to add two parameters to **getDiscount** operation.

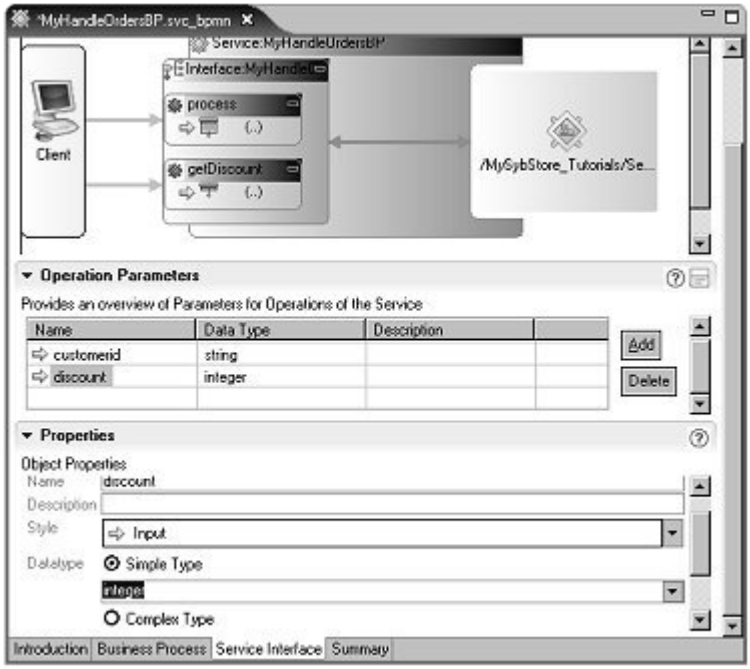
Parameter 1:

- Name – `customerid`
- Style – Input
- Data type – **Simple** integer

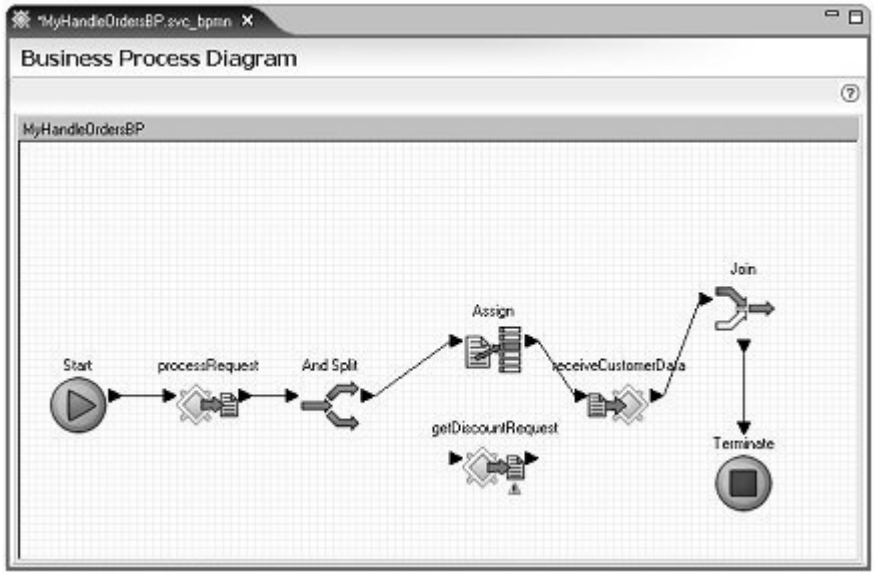
Parameter 2:

- Name – `discount`
- Style – Input

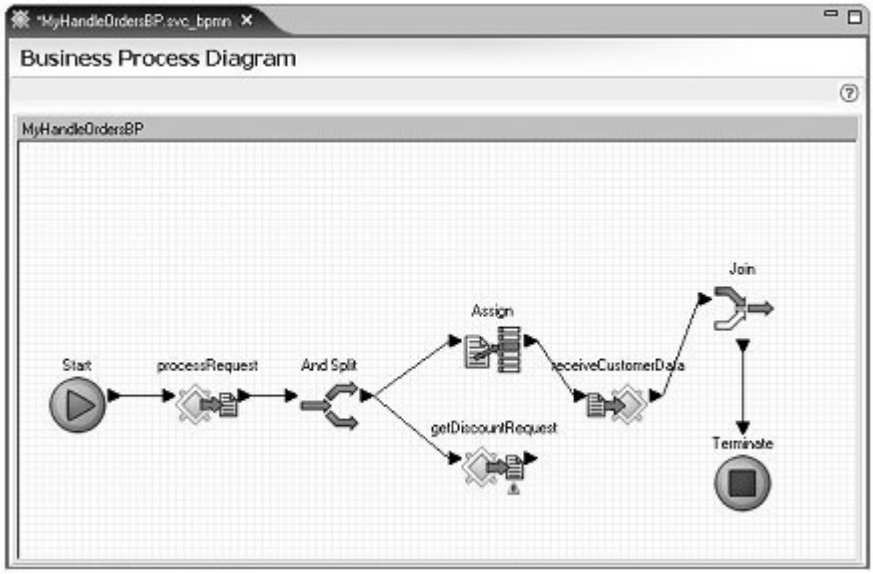
- Data type – **Simple** integer



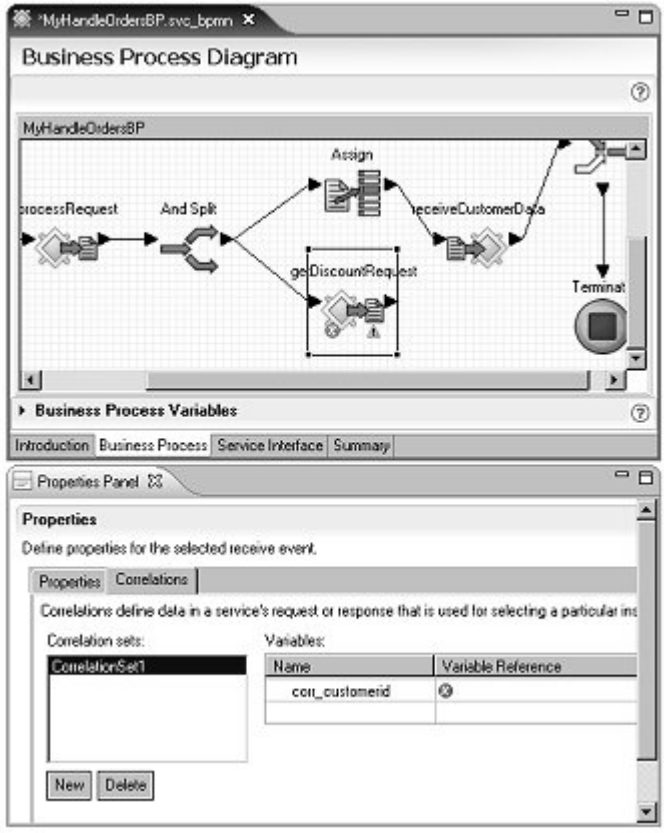
25 Select the **Business Process** tab on the editor. It should show the newly added operation **getDiscountRequest** activity.



26 Move the **getDiscountRequest** activity below the **And Split** activity in the editor canvas. Click the right arrow of **And Split** and drag to and click the left arrow of **getDiscountRequest**.

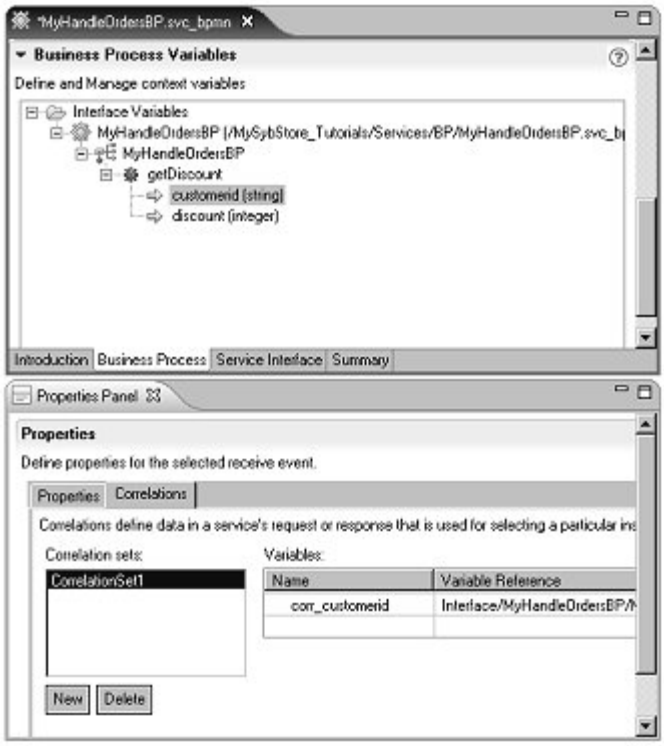


27 In the editor canvas, select **getDiscountRequest**, expand the **Properties Panel**, select the **Correlations** tab, and click **New** below the **Correlation Sets** pane.



28 Expand the **Business Process Variables** pane and expand **Interface Variables/MyHandleOrdersBP/MyHandleOrdersBP/getDiscount** to locate **customerid**.

29 Drag and drop **customerid** to the **Variable Reference** column in the **Variables** table on the **Correlations** tab.



30 Select **File|Save** from the WorkSpace main menu bar and leave the service open in the editor.

Lesson 4: Adding logging activities

In this lesson, add extra nodes to log incoming data to *Jaguar log*.

1 Right-click in the editor canvas and select **Show Tool Palette**.

2 Select the **Activities** category in the **Tool Palette**, then drag and drop an **Assign** activity to the editor canvas next to the **getDiscountRequest** activity.

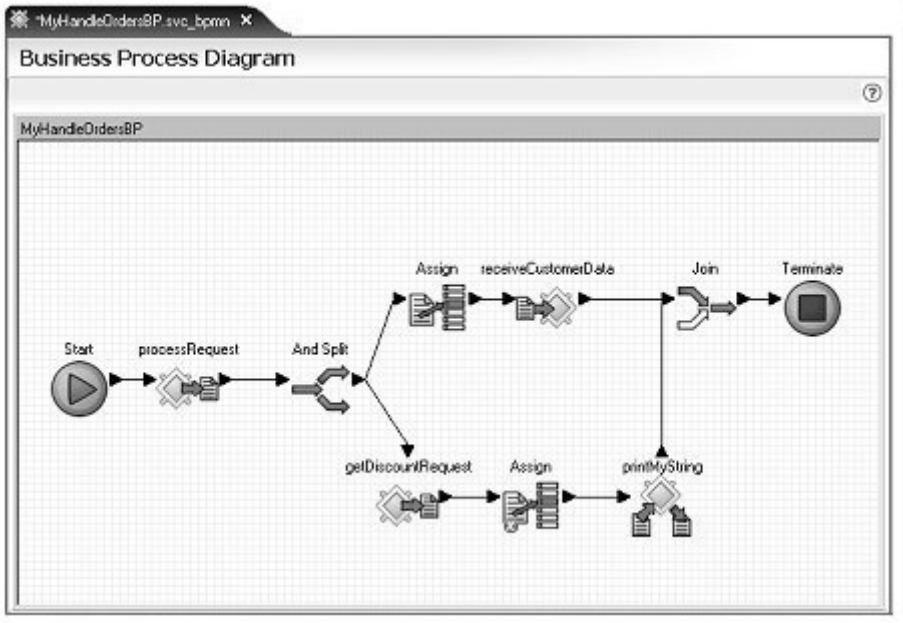
3 In the **Service Explorer**, expand **Private/MyDebugJavaService/debugClass**, select **printMyString** and drag and drop it next to the **Assign** activity on the editor canvas.

4 Connect the activities:

   a Click the right arrow of **getDiscountRequent** and drag to and click the left arrow of the new **Assign**.

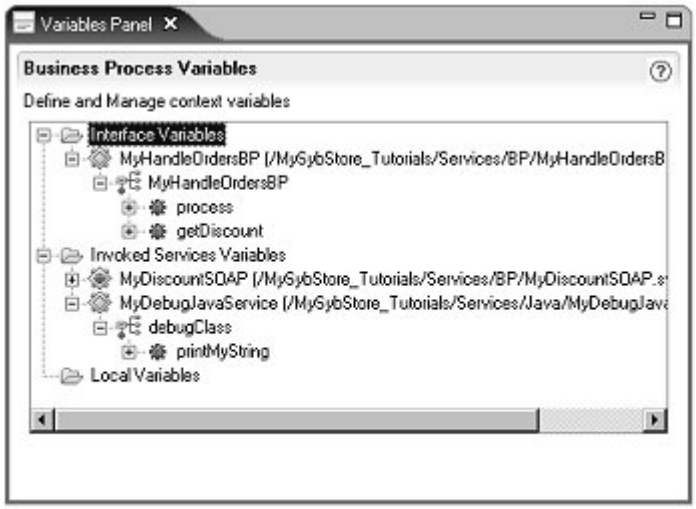
   b Click the right arrow of the new **Assign** and drag to and click the left arrow of **printMyString**.

   c Click the right arrow of **printMyString** and drag to and click the left arrow of **Join**.



5 Expand the **Business Process Variables** view in the **Variables Panel**, expand **Interface Variables/MyHandleOrdersBP/MyHandleOrdersBP/getDiscount** to see the **getDiscount** operation parameters.

6 Expand **Invoked Services Variables/MyDebugJavaSerivce/debugClass** to see the **printMyString** parameter.



7 In the editor canvas, select the new **Assign** activity.

8 Complete two assignments for the first Assign activity so when the business process has successfully executed, a message is logged that indicates the customer ID and the discount.

Expand the **Properties Panel** and complete the **Source** and **Target** for the first **Assign** in the **Assign Overview** table:

a In the **Business Process Variables** view of the **Variables Panel**, drag **customerid** from beneath **Interface Variables/MyHandleOrdersBP/MyHandleOrdersBP/getDiscount** and drop it into the **Source** column.

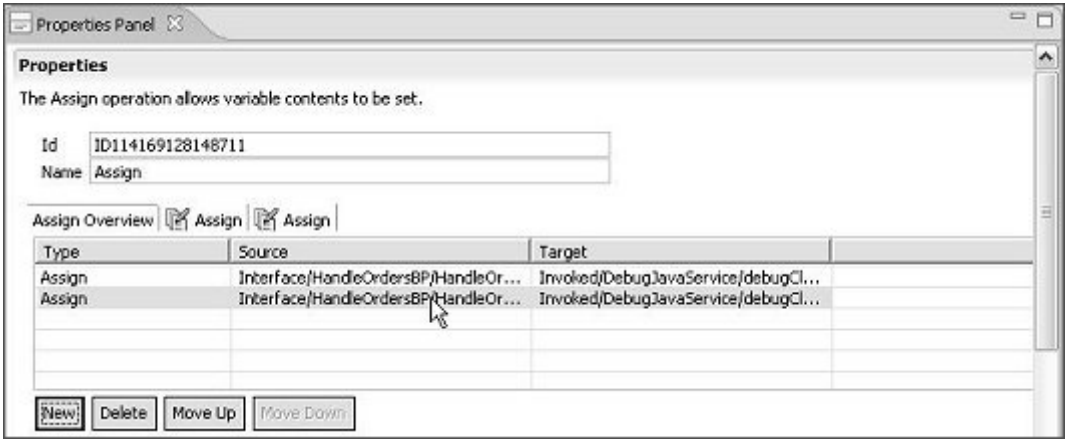
b Drag **label** from beneath **Invoked Service Variables/MyDiscountSOAP/MyDebugJavaService/debugClass/printMyString** and drop it into **Target** column.

9 Click **New** below the **Assign Overview** table in the **Properties Panel** to add one more assignment row.

10 Complete the **Source** and **Target** properties for the second assignment:

a In the **Business Process Variables** view of the **Variables Panel**, drag **discount** from beneath **getDiscount** and drop it into the second Assign's **Source** column.

b Drag **variable** from beneath **printMyString** and drop it into the second Assign's **Target** column.



11 Select **File|Save** from the WorkSpace main menu bar.

12 Use the instructions in "Packaging, deploying, and testing a business process service" on page 143 to create a package and deploy the **MyHandleOrdersBP** service.

Note Unwired Orchestrator must be running and there must be a connection to the server from MyServiceContainer in WorkSpace.

---

13 In **Enterprise Explorer**, expand **Service Containers/MyServiceContainer/Packages/MyHandleOrdersBP/Services** and locate the **MyHandleOrdersBP** package.

14 In the **Enterprise Explorer**, right-click **MyHandleOrdersBP** and select **Save WSDL** from the context menu. When the **Save WSDL Wizard** opens, select **MySybStore_Tutorials/Services/BP** as the parent folder, confirm the **File Name** is **MyHandleOrdersBP,** and click **Finish**.

15 In the **WorkSpace Navigator**, expand **MySybStore_Tutorials/Services/BP,** right-click **MyHandleOrdersBP.wsdl**, and select **Create SOAP Service** from the context menu.

16 When the **New Service Wizard** opens, select **MySybStore_Tutorials/Services/BP** as the parent folder, enter `MyProcessOrderSOAP` for the **File Name** of the service, and click **Next**.

17 In the **Service Summary** window, click **Next**.

18 In the **Service Endpoint Creation** window, select the option **Yes, Create An Endpoint Now**, and click **Next**.

19 In the **Endpoint Name** window, accept the default **Name** `endpoint` and click **Next**.

20 In the **Connection Properties** window, click **Finish**.

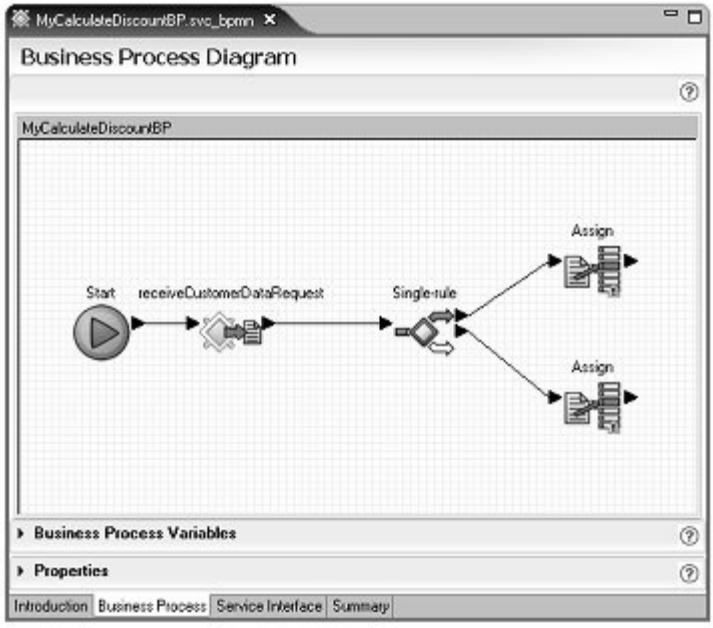
21 The SOAP service is created and opens in the SOAP Service Editor. Select **File|Close**.

Lesson 5: Sending a correlated response to the initiating business process

Return to the MyCalculateDiscountBP service to finish its implementation and send the correlated response to MyHandleOrdersBP.

1 Open the **Service Development** perspective.

2 In the **WorkSpace Navigator**, expand **MySybStore_Tutorials/Services/BP** and double-click **MyCalculateDiscountBP.svc_bpmn** to open it in the Business Process Service Editor.



3 Right-click in the editor and select **Show Tool Palette** from the context menu.

4 In the **Tool Palette**, drag and drop these activities to the editor canvas:

(Tool Palette category | [Logic|Activities])

- **Logic | Join** to the right of the two **Assigns**
- **Activities | Delay** to the right of the new **Join**
- **Activities | Assign** to the right of the **Delay** activity

5 In the **Service Explorer**, expand **Public/MyProcessOrderSOAP/MyHandleOrdersBP**, and drag and drop the **getDiscount** operation on to the editor canvas to the right of the **Assign** on the far right of the canvas.

6 Connect the activities on the editor canvas, rearranging the objects as necessary.

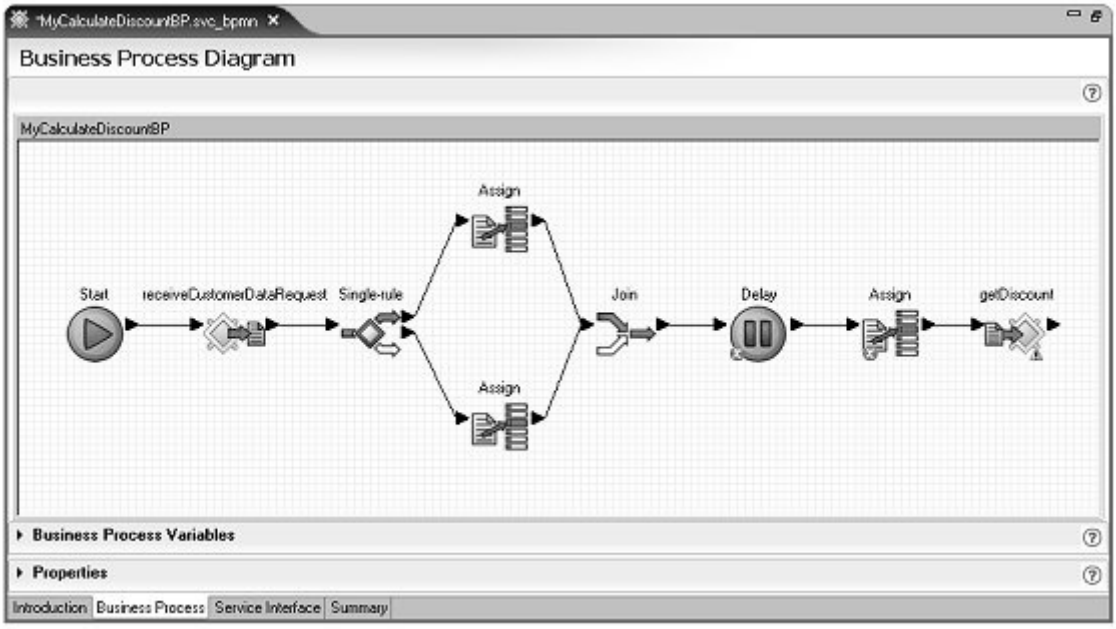
a Click the right arrow of the top **Assign** and drag to and click the left arrow of **Join**.

b Click the right arrow of the bottom **Assign** and drag to and click the left arrow of **Join**.

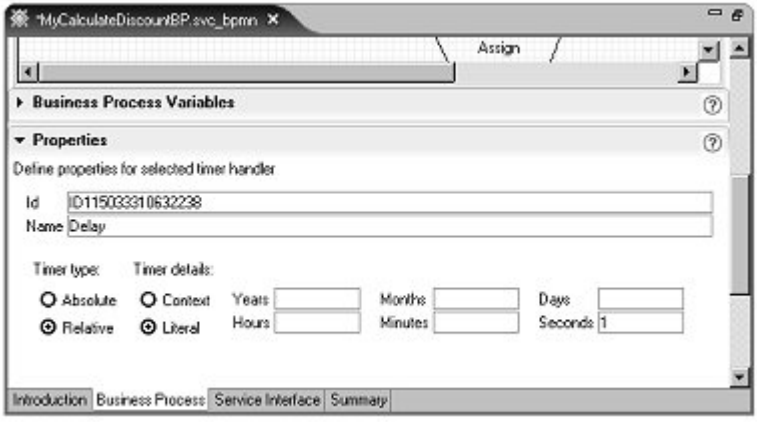
c Click the right arrow of **Join** and drag to and click the left arrow of **Delay**.

d Click the right arrow of **Delay** and drag to and click the left arrow of the last **Assign**.

e Click the right arrow of the last **Assign** and drag to and click the left arrow of **getDiscount**.



7 Select **Delay** in the editor canvas and expand the **Properties** pane. Set the **Timer Type** to **Relative**, and set the **Timer Details** to **Literal** and `1` **Second**.

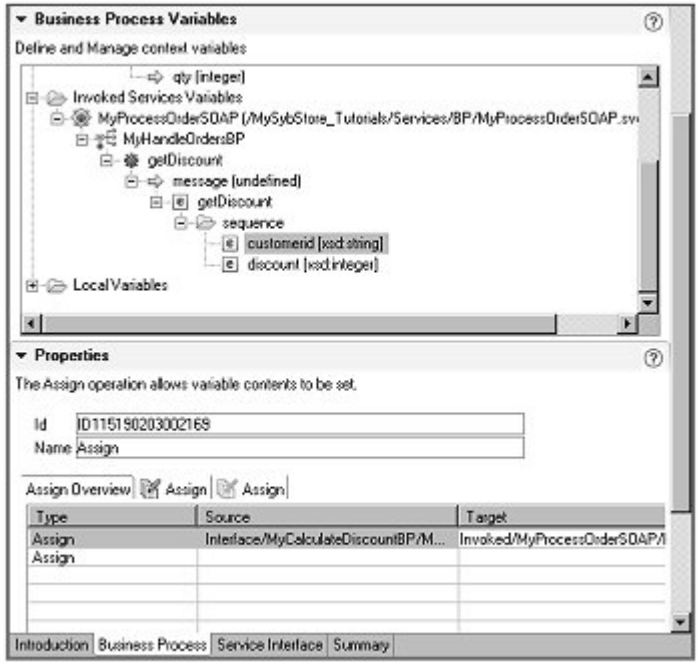


8 Select the **Assign** activity on the far right of the editor canvas, expand the **Properties** pane, and click **New** to add one more assignment row. There are now two **Assigns** in the **Assign Overview** table.

9 Expand the **Business Process Variables** pane, and specify each assignment's **Source** and **Target** value in the **Property** pane as follows:

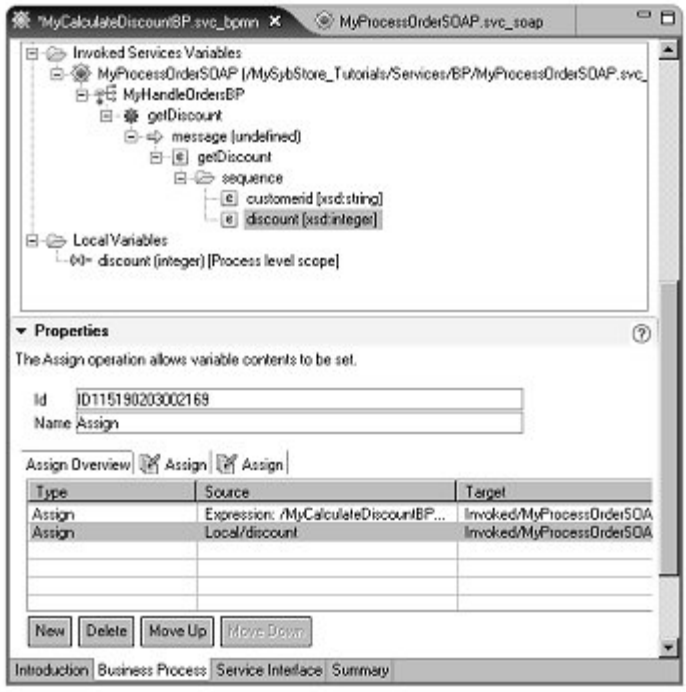
a Drag **customerid** from **Interface Variables/MyCalculateDiscountBP/MyCalculateDiscountBP/receiveCustomerData** to the **Source** column of the first assignment in the **Assign Overview** table.

b Drag **customerid** from **Invoked Services Variables/MyProcessOrderSOAP/MyHandleOrdersBP/getDiscount/message/getDiscount/sequence** to the **Target** column of the first assignment.



c In the **Business Process Variables** pane, expand the **Local Variables** folder and drag **discount** to the **Source** column of the second assignment in the **Assign Overview** table.

d Drag **discount** from **Invoked Services Variables/MyProcessOrderSOAP/MyHandleOrdersBP/getDiscount/message/getDiscount/sequence** to the **Target** column of the second assignment.



10 Select **File|Save** from the WorkSpace main menu bar.

11 To resynchronize the package because you added a new SOAP service operation to the business process, right-click **MyCalculateDiscountBP.svc_bpmn** again and select **Create Sybase Services Package Profile** from the context menu. Answer yes when you are asked if you want to overwrite the existing file.

12 Right-click **MyCalculateDiscountBP.svcpkgdef**, and select **Build Package** from the context menu. When you are prompted to overwrite an existing file, click **Yes to All**. When a message states that the package was built successfully, click **OK**.

13 To deploy the business process, right-click **MyCalculateDiscountBP.svcpkgdef**, and select **Deploy Package**.

14 When the **Select Target Server** dialog box opens, select **MyServiceContainer** and click **OK**. When you are prompted to overwrite an existing file, click **Yes to All**. If a message warns you about existing packages of the same name already existing, click **OK**.

15 When you see a message stating that the package was successfully deployed, click **OK**.

16 Close the **Console** window and select **File|Close** from the WorkSpace main menu to close any editor windows.

Upon successful execution, MyHandleOrdersBP logs a message in *Jaguar.log* that prints the customer ID and the discount given to that customer (10 or 0, depending on the value you enter for input quantity.

Executing the business process results in a call made to the receiveCustomerData operation of the MyDiscountSOAP service, which in turn calls the getDiscount operation on the waiting MyProcessOrdersSOAP service (MyHandleOrdersBP). The call to getDiscount is correlated, based on the value of customer ID, then sent to the waiting service instance.

Congratulations! You have completed the Process Orchestration tutorials.

CHAPTER 4

Unwired Orchestrator Logging Tutorials

The Unwired Orchestrator logging tutorial teaches you how to generate verbose logging data for a service in the SybStore tutorial project. The tutorial also show you how to import the Unwired Orchestrator log file, view its contents, and apply filters to control the information that displays.

Topic	Page
Overview	203
Using Unwired Orchestrator logging	204

Overview

Sybase WorkSpace allows you to use the Eclipse Tracing and Profiling Tools Project (TPTP) framework to import, view, and filter the contents of the Sybase Unwired Orchestrator log file. This log file observes the Eclipse “Common Base Event” format. This format is required to view and filter the log contents in the Profiling and Logging perspective Log View within Sybase WorkSpace.

For more information about Unwired Orchestrator logging capabilities, see the WorkSpace online help topic *Sybase WorkSpace Server Administration|Sybase Unwired Orchestrator 5.1*.

Prerequisites

Before you begin this tutorial, complete all of the procedures in Chapter 1, “Introduction, Installation, and Setup.”

Using Unwired Orchestrator logging

This tutorial teaches you how to use the Unwired Orchestrator logging functionality.

This tutorial contains of:

Lesson 1: Setting the logging level and deploying a service
Lesson 2: Executing a service to generate log data
Lesson 3: Importing the Unwired Orchestrator log file
Lesson 4: Reviewing the Unwired Orchestrator log file
Lesson 5: Viewing selected portions of the log file

Lesson 1: Setting the logging level and deploying a service

In this lesson, you will increase the verbosity of messages written to the Unwired Orchestrator log file and deploy a business process service, in preparation for executing the service to generate logging data.

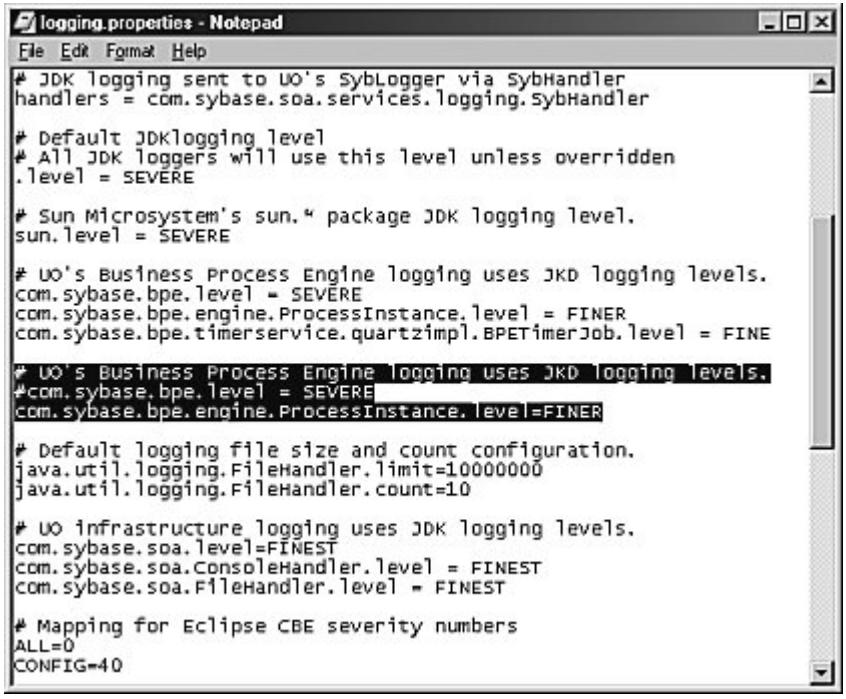
1 In the Windows **File Explorer,** navigate to *%WORKSPACE_DIR%\WorkSpace\DevRuntimes\EAServer\bin*.

2 Open the *logging.properties* file in a text editor and replace these lines:

```
# UO's Business Process Engine logging uses JKD logging levels.
com.sybase.bpe.level = SEVERE
```

with these lines:

```
# UO's Business Process Engine logging uses JKD logging levels.
#com.sybase.bpe.level = SEVERE
com.sybase.bpe.engine.ProcessInstance.level=FINER
```



3 If Unwired Orchestrator is running, select **Start|Programs|Sybase|Sybase WorkSpace|UO 5.1|Stop UO**.

4 Restart Unwired Orchestrator to initialize the server with the new logging level. Select **Start|Programs|Sybase|Sybase WorkSpace|UO 5.1|Start UO**.

5 Select **Start|Programs|Sybase|Sybase WorkSpace|Sybase WorkSpace 1.5**.

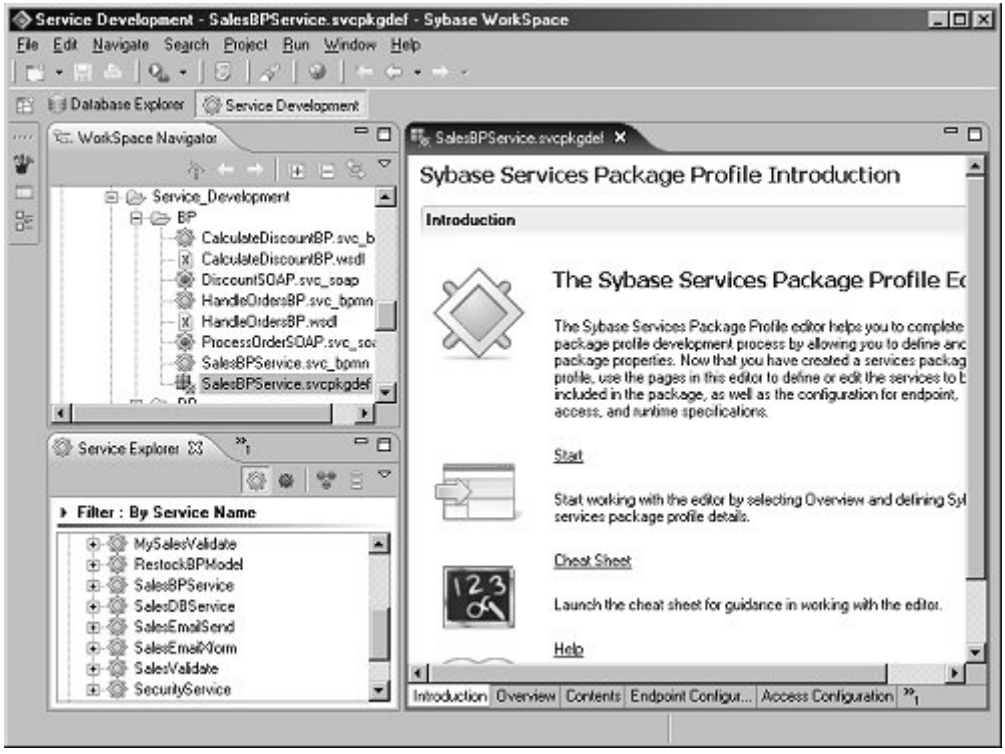
6 Select **Window|Open Perspective|Service Development** on the main menu bar.

7 In the **WorkSpace Navigator**, expand **MySybStore_Tutorial/Tutorial_Resources/Service_Development/BP**.

8 Create a service package from an existing business process service.

Right-click the **SalesBPService.svc_bpmn** file and select **Create Sybase Service Packages Profile** from the context menu.

The package definition is created (*SalesBPService.svcpkgdef*) and opens in the Sybase Services Package Profile Editor.



9 Select the **Runtime Container Configuration** tab and set the **Log Level** to **FINE (minimum debug logging)**.

10 Select **File|Save** from the WorkSpace main menu bar, then select **File|Close**.

11 If Unwired Orchestrator is running and a connection is already established, go to step 12. If not, use the instructions in “Starting and connecting to the Unwired Orchestrator server” on page 19 to start and connect to the server.

12 Deploy the service:

   a In the **WorkSpace Navigator**, right-click the **SalesBPService.svcpkgdef** file.

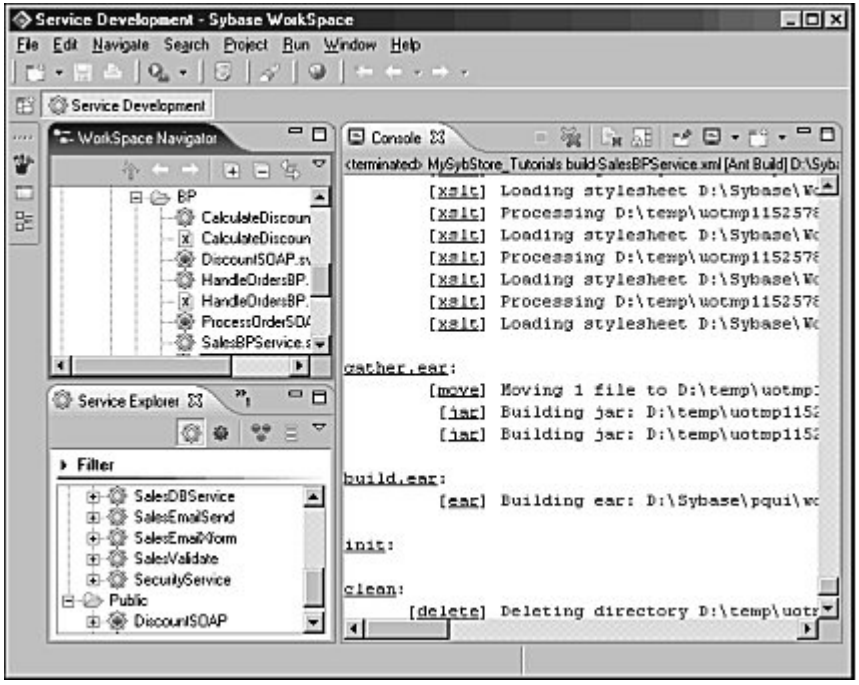
   b Select **Deploy Package** from the context menu.

   c When the **Select Service Container** dialog box opens, choose **MyServiceContainer** and click **OK**.

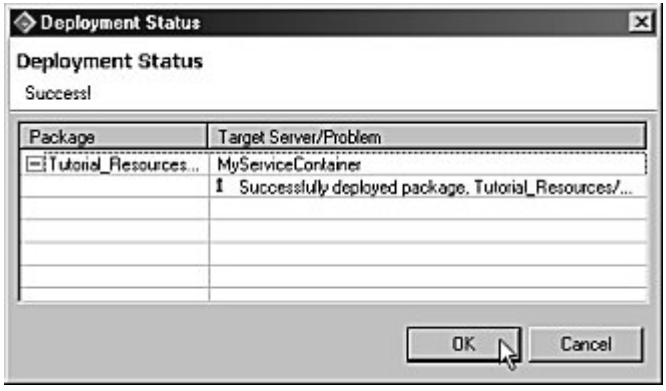
   d If you see a **Pre-Deploy Check Status** message that states a problem may exist because the package already exists, click **OK**.

This step deploys the package, which can take a few minutes, with the business process service and its dependent services to the service container called MyServiceContainer.

You see a progress window as the service deploys and a variety of messages in the WorkSpace **Console** window.



13 When you see the **Deployment Status** window indicating that the deployment was successful, click **OK** to close the window.



14 Close the **Console** window by clicking the "**X**" on the window's title tab.

You have set the logging level for the Unwired Orchestrator server to a verbose level, and deployed a business process service in preparation for executing it and generating logging data to view. Because the SybStore database is not running, the service that you deployed will not execute completely.

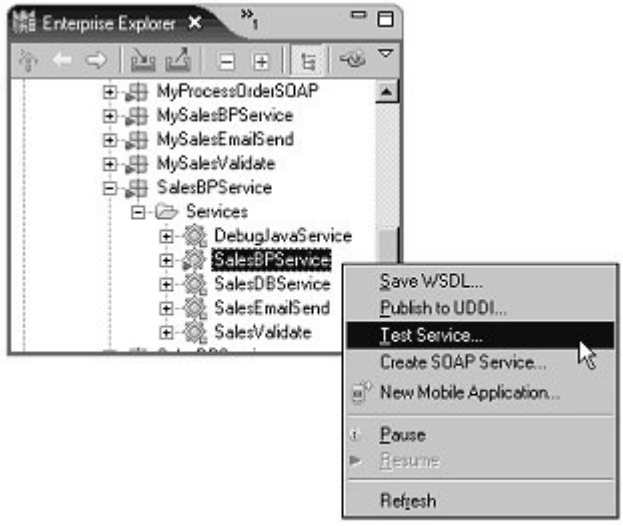
You see some messages in the log file that indicate successful execution, and some messages that indicate errors. The mixture of message severities may be useful as you complete the remaining lessons of this tutorial.

Lesson 2: Executing a service to generate log data

In this lesson, you will execute the business process service SalesBPService to generate data in the Unwired Orchestrator log file.

1 Select **Window|Show View|Enterprise Explorer** on the main menu bar.

2 In the **Enterprise Explorer**, expand **Service Containers/MyServiceContainer/Packages/SalesBPService/Services**, right-click the **SalesBPService** service, and select **Test Service** from the context menu.



3 When the **Service Testing Wizard** opens, select **Dialog Service Test Wizard** and click **Next**.

4 In the **Options** window, accept the default entries and click **Next**.

5 In the **Select a Method to Test** window, choose `SalesDetailResponse manageInventory(SalesDetail inputSalesItem)` and click **Next**.

6 When the **SOAP Request and Response** window opens, click **Load Request Message**.

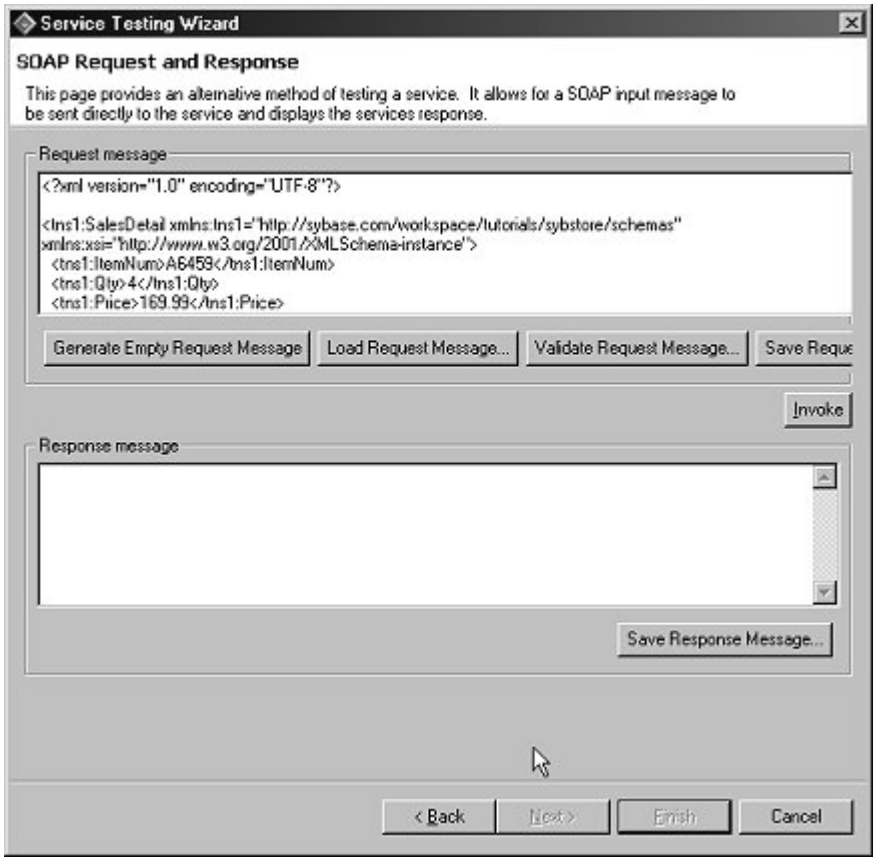
7 In the **Select a SOAP Request** dialog box, navigate to:

x`\<`*your_workspace*`>\MySybStore_Tutorials\Services\BP\TestData\`

where *x:/<your_workspace>* is where your personal WorkSpace files are stored.

8 Select **SalesProcessingBPService.xml** and click **Open**.

The request message you selected displays in the window.



9 Click **Invoke** to invoke a response message.

10 Scroll through the text that appears in the **Response Message** pane. You should see these lines:

```
<soapenv:Body>
  <ns1:SalesDetailResponse xmlns:ns1="http://sybase.com/workspace
  /tutorials/sybstore/schemas">Invalid Sales Item
  </ns1:SalesDetailResponse>
</soapenv:Body>
```

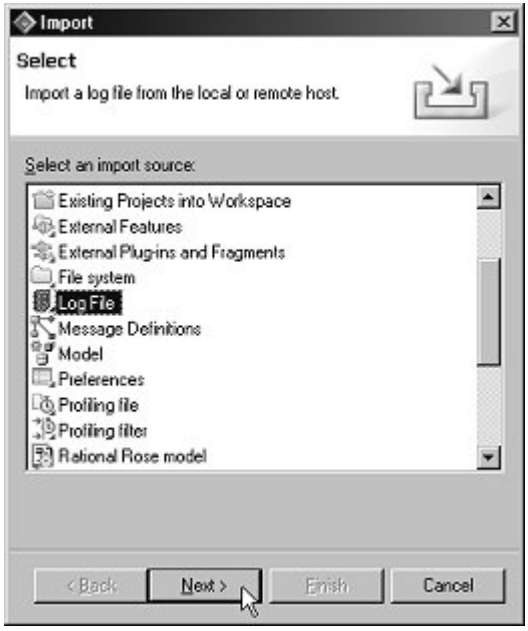
11 Click **Finish** to close the testing window.

You have tested a business process service that executes some activities successfully and notifies you that there is an invalid sales item. You are now ready to view the contents of the Unwired Orchestrator log file.

Lesson 3: Importing the Unwired Orchestrator log file

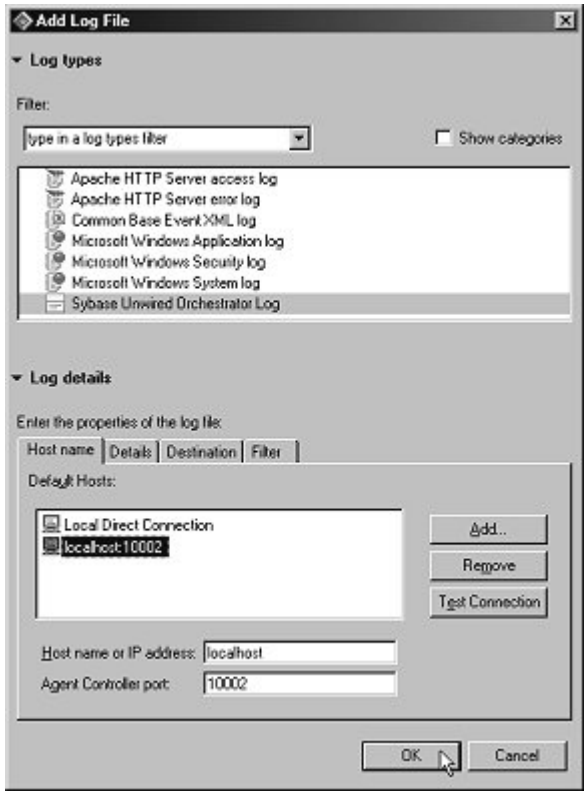
1 In the **WorkSpace Navigator**, select **File|Import** from the main menu bar.

2 When the **Import** selection window opens, select **Log File** and click **Next**.



The Import Log File window opens. Specify the location of the log file you want to import.

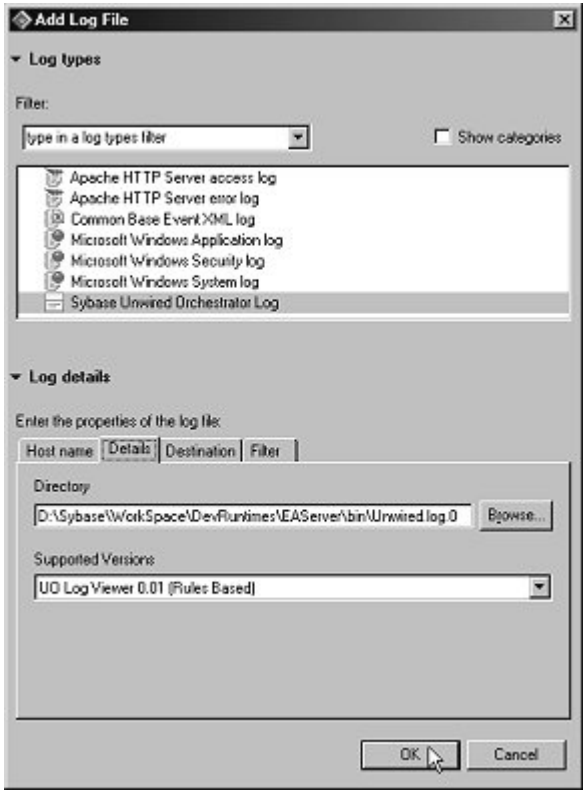
3 Click **Add**. The Add Log File window opens.



4 In the **Log Types** section, select **Sybase Unwired Orchestrator Log**.

5 In the **Log Details** section, select **localhost:10002** or **Local Direct Connection**; either host will work.

6 In the **Log Details** section, select the **Details** tab.

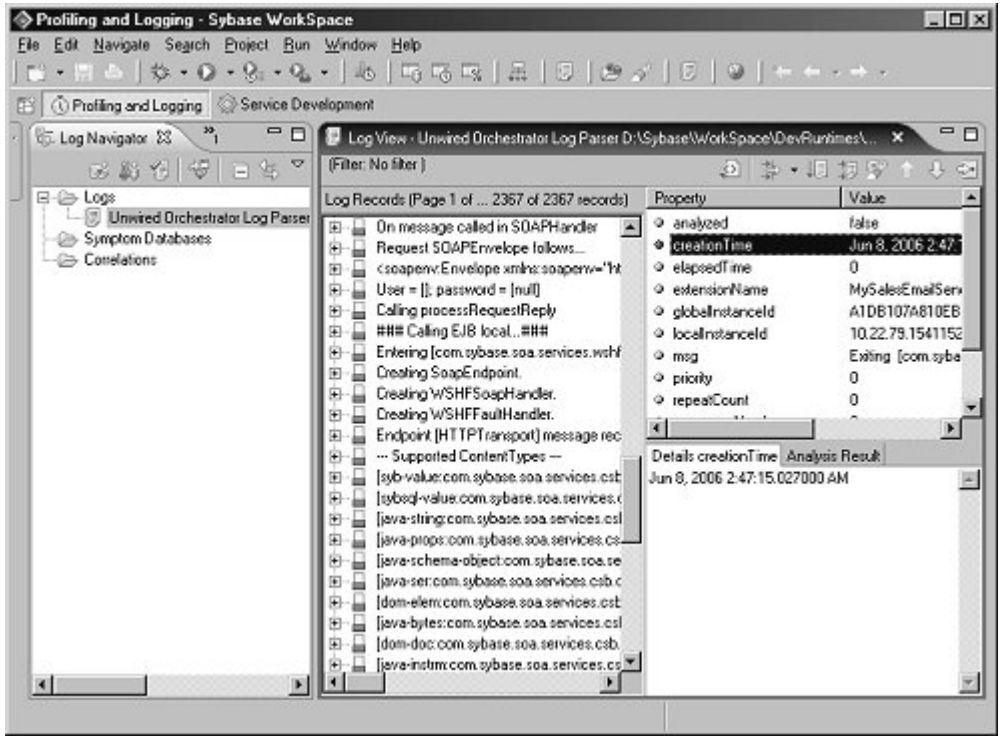


Click **Browse**. When the **Open** file dialog box appears, navigate to *%WORKSPACE_DIR%\DevRuntimes\EAServer\bin*, where *%WORKSPACE_DIR%* is the location where Sybase WorkSpace is installed.

7 Select the **Unwired.log.0** file, click **Open**, then click **OK** to close the **Add Log File** window.

8 When you return to the **Import Log File** dialog box, you see the name of the log file you chose to import. Click **Finish**.

9 When you are prompted to switch to the **Profiling and Logging** perspective, click **Yes**. The log file content displays in the Log View.

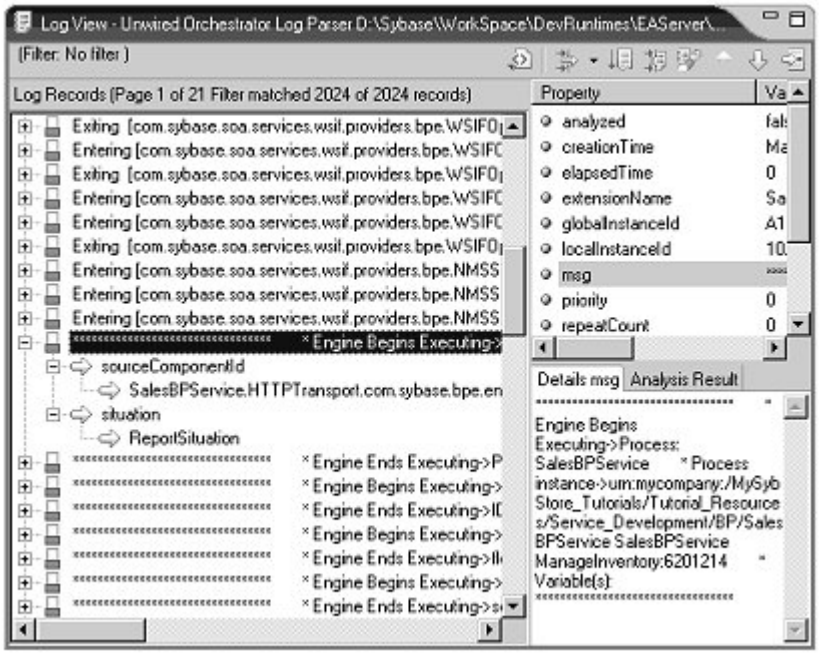


You have imported the Unwired Orchestrator log file into WorkSpace and are ready to examine its contents.

Lesson 4: Reviewing the Unwired Orchestrator log file

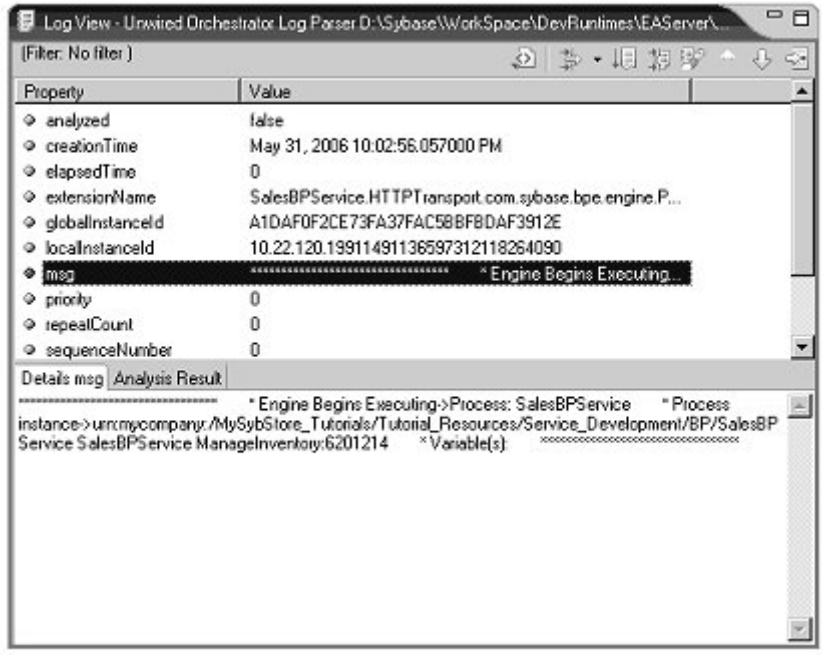
In this lesson, you will focus on a particular message in the log file and see how you can view the individual message components in the WorkSpace Log View.

1 In the WorkSpace **Log View**, look for the message where Unwired Orchestrator begins executing the business process service, and expand the tree view beneath it by clicking the elements preceded by a plus sign.



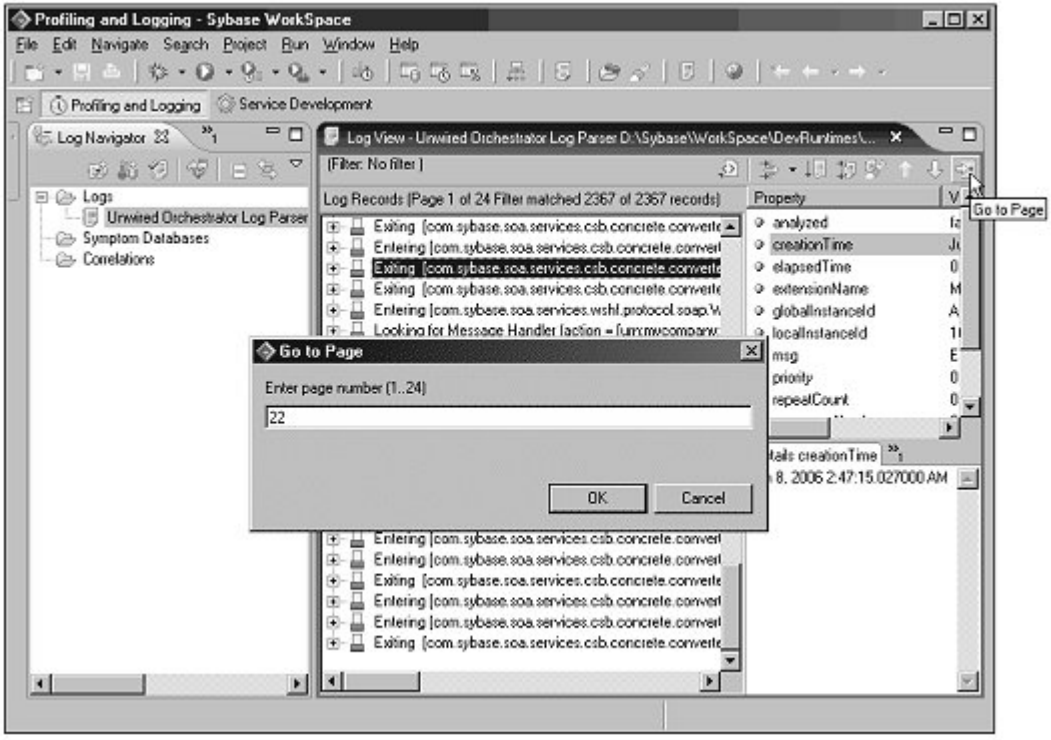
Each tree node in the Log Records Pane represents a different page in the log file. Depending on what part of the tree you highlight in the Log Records pane, you see different information in the Property pane.

2 In the **Property** pane, highlight the "msg" property. The property has a truncated value in the **Property** pane, but the complete value displays in the **Details Msg** pane.



3 Expand another tree node to view a different page in the log file.

4 Click the **Go To Page** icon to search through the log by page number.



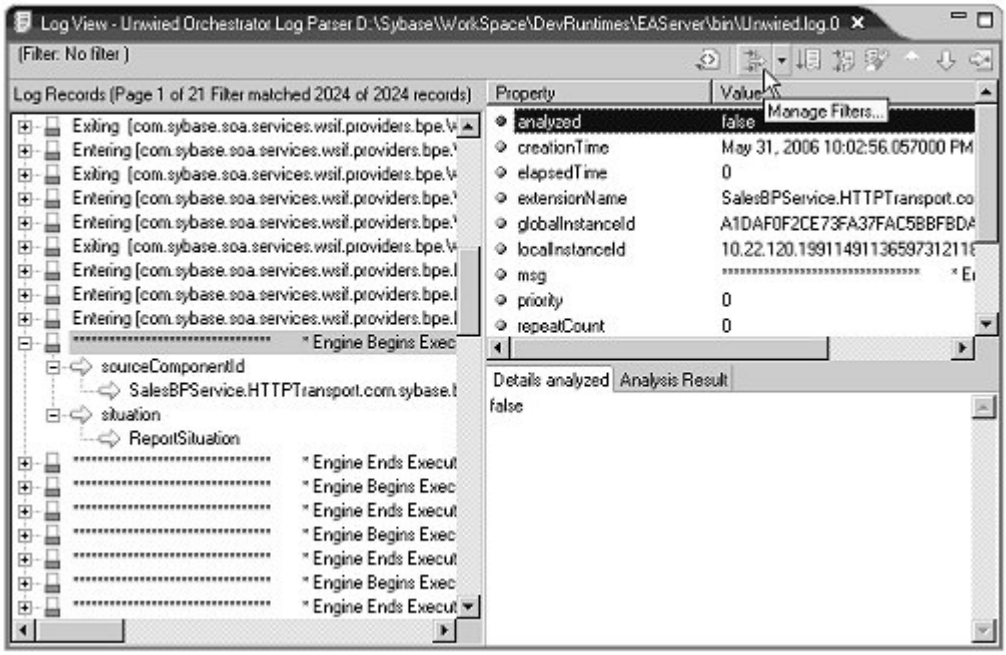
5 Enter the page number and click **OK**. Try selecting various log file messages, expanding the tree view, and viewing the details in the various panes.

You have seen the different kinds of information you can view about a particular log file message, and learned how to navigate to different log file pages.

Lesson 5: Viewing selected portions of the log file

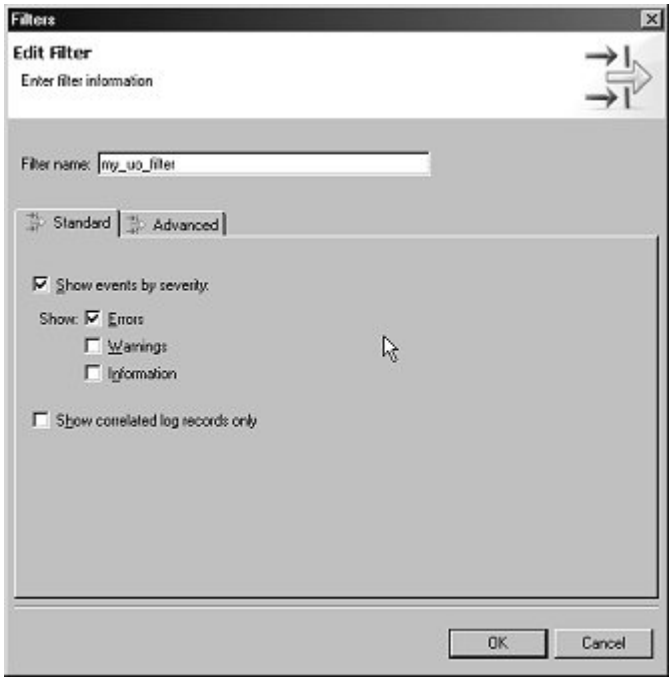
In this lesson, you create a filter to view only the log file's exception messages.

1 Click the **Manage Filters** icon on the **Log View** toolbar to create a new log file filter.



2 When the **Filters** dialog box opens, select **Log** as the type of filter to add and click **OK**.

3 In the **Edit Filters** dialog box, enter a name for the filter, such as `my_uo_filter`.



4 Select the **Advanced** tab and click **Add** to add a new filter.

5 In the **Add Filter Property** dialog box, complete these options:

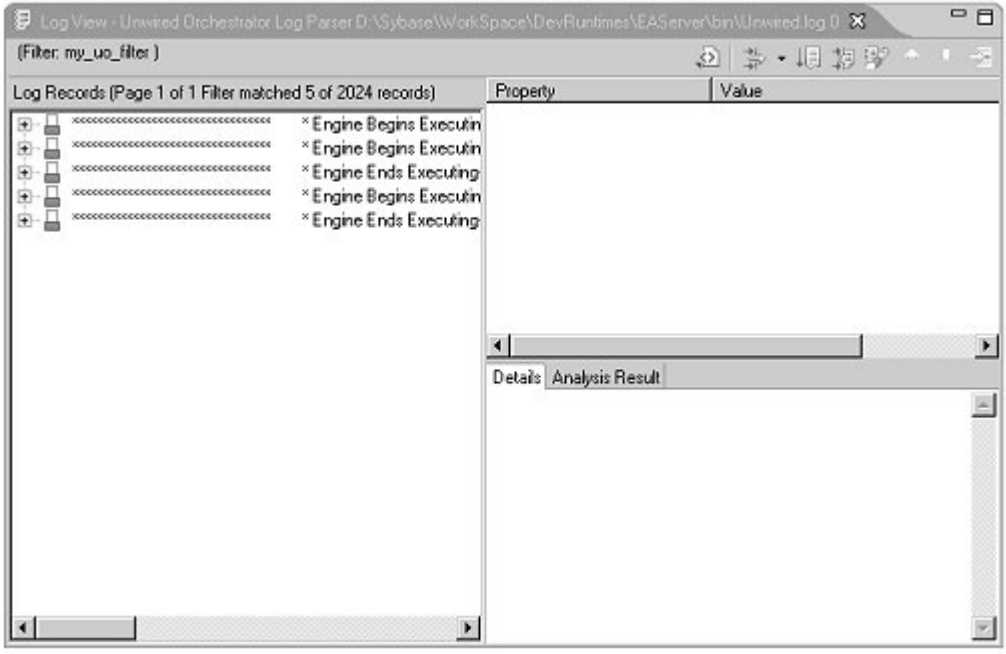
- Attribute – select **Msg** from the drop-down list.
- Operator – select **like** from the drop-down list.
- Value – enter `*exception*`.

Click **OK** to close the **Add Filter Property** dialog box.

6 Click **OK** to close the **Edit Filter** dialog box.

7 Click **OK** to close the **Filters Add/Edit/Remove** list box.

Now you see only messages that include the text “exception.”



8 Expand the tree view beneath one of the exception messages to view that message's details.

9 Turn off the filter to view the complete contents of the log again.

You have created a filter to view only the log file's exception messages, looked at the details of an exception message, then turned the filter off.

Congratulations! You have completed the logging tutorial.

CHAPTER 5

Cleaning up the Sybase WorkSpace environment

Use these instructions to disconnect from WorkSpace servers, stop WorkSpace components, remove the tutorial project, and clean up the WorkSpace environment.

Topic	Page
Closing active connections	223
Deleting the tutorial project	224
Recreating the tutorial project	224

Closing active connections

1. In the WorkSpace **Enterprise Explorer**, expand the **Databases** folder.
2. Right-click the **MySybStore** connection profile and select **Disconnect** from the context menu.
3. To shutdown the tutorial database, in the Windows system tray, right-click the Adaptive Service Anywhere (SQL) icon and select **Exit**.
4. In the WorkSpace **Enterprise Explorer**, expand the **Service Containers** folder.
5. Right-click the **MyServiceContainer** connection profile and select **Disconnect** from the context menu.
6. Outside of WorkSpace, select **Start|Programs|Sybase|Sybase WorkSpace|UO 5.1|Stop UO**.

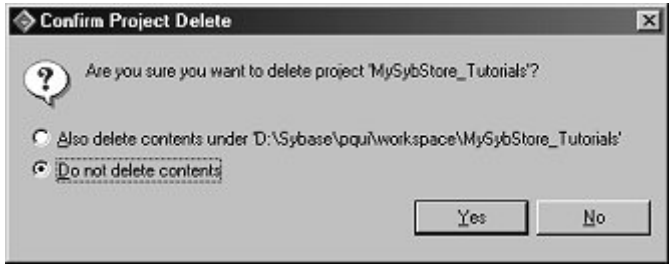
Deleting the tutorial project

1 Select **Window|Open Perspective|Other**, choose **Service Development** from the **Select Perspective** dialog box, and click **OK**.

2 In the **WorkSpace Navigator**, right-click **MySybStore_Tutorial** and select **Delete** from the context menu.

3 When asked to confirm the deletion, select:

- **Also Delete Contents Under**... – *<your_personal_WorkSpace_path>* – to remove the tutorial source, generated, and user-created files from WorkSpace and from your computer's hard drive.
- **Do Not Delete Contents** – to remove the project from WorkSpace but to retain all tutorial-related files on your computer's hard drive.



4 Click **OK**. The project is deleted.

5 Select **File|Exit** from the main menu bar to shut down WorkSpace.

Recreating the tutorial project

Use these instructions to recreate the SybStore tutorial project when the projects have been deleted from WorkSpace, but the project files remain on your computer's hard drive.

1 Select **Window|Open Perspective|Other**, choose **Service Development** from the **Select Perspective** dialog box, and click **OK**.

2 Select **File|New|Project** from the WorkSpace main menu.

3 When the **New Project** wizard opens, select **Sybase|Sybase WorkSpace Project** and click **Next**.

4 When the **Sybase WorkSpace Project** window complete these options:

   - Project Name – enter `MySybStore_Tutorials`.
   - Project Contents – select:
     - **Use Default** – to recreate the project in the default personal WorkSpace directory if that is where you created the project originally.
     - **Directory** – if you created the project is a location other than your default personal WorkSpace directory. Unselect **Use Default**, then click **Browse** and navigate to where the project's files are located.

5 Click **Finish**. The project is created with all of the source, generated, and user-defined files that existed when you deleted the project.