SeeBeyond ICAN Suite

eXchange Integrator User's Guide

Release 5.0.4

SEEBEYOND®

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces you to this guide, its general purpose and scope, and its organization. It also provides sources of related documentation and information.

In this chapter

1.1 Overview

The User's Guide provides instructions and background information for all users of the eXchange Integrator application. This guide is designed for managers, system administrators, and others who use eXchange Integrator.

The purpose of this guide is to help you do the following:

- Understand the nature of eXchange.
- Understand the function of eXchange.
- Understand the relationship of eXchange to other components of the SeeBeyond Integrated Composite Application Network (ICAN) Suite.
- Learn about the eXchange components and editors and how to use them in your environment.

1.2 Contents of This Guide

This guide is arranged as follows:

- **Chapter 1, "Introduction"** provides an overview of this document's purpose, contents, writing conventions, and supported documents.
- **Chapter 2, "Overview"** discusses general features and architecture of eXchange.
- **Chapter 3, "Installing eXchange"** provides step-by-step instructions for installing the eXchange product and setting it up for use.
- **Chapter 5, "Using eXchange in Enterprise Designer"** provides step-by-step procedures for working with eXchange at design time and deploying projects.
- **Chapter 6, "Designing B2B Protocols"** provides step-by-step procedures for designing and deploying B2B protocols using eXchange Protocol Designer.
- **Chapter 7, "Exception Handling"** explains the use of B2B protocols for handling exceptions.
- **Chapter 8, "Using eXchange Web Facilities"** provides step-by-step procedures for working with eXchange's Web-based GUIs—Partner Manager (ePM) and Message Tracking—as well as using Monitor for B2B protocols.
- **Chapter 9, "Implementation Scenario: AS2"** provides step-by-step for creating implementation scenarios that provide a sample of how eXchange can be used to achieve B2B solutions.
- **Appendix A**, **"Method Palette"** lists and describes the tools available to you in eXchange Protocol Designer.

1.3 Writing Conventions

The following writing conventions are observed throughout this document.

Table 1 Writing Conventions

Text	Convention	Example
Names of buttons, files, menus and menu items, icons, parameters, variables, methods, and objects	**Bold** text	▪ Click **OK** to save and close. ▪ Select the **logicalhost.exe** file. ▪ On the **File** menu, click **Exit**. ▪ Enter the **timeout** value. ▪ Use the **getClassName()** method.
Command-line arguments, code samples	`Fixed` font. Variables are shown in ***bold italic***.	`bootstrap -p` ***password***
Hypertext links	**Blue** text	▪ For more information, see **"Writing Conventions" on page 15**. ▪ **http://www.seebeyond.com**

Additional Conventions

Windows Systems

For the purposes of this guide, references to "Windows" will apply to Microsoft Windows Server 2003, Windows XP, and Windows 2000.

Path Name Separator

This guide uses the backslash ("\") as the separator within path names. If you are working on a UNIX system, please make the appropriate substitutions.

1.4 Supporting Documents

For more information about eXchange and the ICAN Suite, refer to the following:

Title	Filename
SeeBeyond ICAN Site Installation Guide	ICAN_Install_Guide.pdf
SeeBeyond ICAN Suite Primer	Primer.pdf
SeeBeyond ICAN Suite Deployment Guide	Deployment_Guide.pdf
eGate Integrator Release Notes	eGate_Release_Notes.pdf
eGate Integrator Tutorial	eGate_Tutorial.pdf
eGate Integrator User's Guide	eGate_User_Guide.pdf
eGate Integrator System Administration Guide	Sys_Admin_Guide.pdf
eGate Integrator JMS Reference Guide	eGate_JMS_Reference.pdf
Oracle eWay Intelligent Adapter User's Guide	Oracle_eWay.pdf
HTTP(S) eWay Intelligent Adapter User's Guide	HTTPS_eWay.pdf
AS2 Protocol Manager Composite Application User's Guide	eXchange_Protocol_Guide_AS2.pdf
ebXML Protocol Manager Composite Application User's Guide	eXchange_Protocol_Guide_ebXML.pdf
X12 OTD Library User's Guide	X12_OTD_Library.pdf
HIPAA OTD Library User's Guide	HIPAA_OTD_Library.pdf
UN/EDIFACT OTD Library User's Guide	EDIFACT_OTD_Library.pdf
Readme for ICAN 5.0.4	Readme.txt

1.5 Online Documents

The documentation for the SeeBeyond ICAN Suite is distributed as a collection of online documents. These documents are viewable with the Acrobat Reader application from Adobe Systems. Acrobat Reader can be downloaded from:

http://www.adobe.com

1.6 The SeeBeyond Web Site

The SeeBeyond Web site is your best source for up-to-the-minute product news and technical support information. The site's URL is:

http://www.seebeyond.com

Chapter 2

Overview

This chapter provides a general overview of eXchange and its place in the ICAN Suite, including system descriptions, general operation, and basic features.

In this chapter

2.1 Summary of Features

eXchange provides an open B2B protocol framework to support standard EDI and B2B business protocols and enveloping protocols. Not only does it support existing standard protocols, with an extensive set of prebuilt business protocol pipelines, it also provides the tools and framework to create and adopt new protocols and to build custom pipelines.

B2B modeling semantics are exposed so that business rules can be added and tailored to address the particular needs of each eBusiness challenge. The tight integration with the rest of the ICAN Suite provides validation, logging, and reporting capabilities, and because each logical step within any business rule is accessible anywhere along the entire business pipeline, the design tools provide complete end-to-end visibility.

The trading partner management facility, eXchange Partner Manager (ePM), is provided via a Web interface. For easy interoperability, trading partners can be configured by importing Collaboration Protocol Agreements (CPAs); or trading partner profiles can be configured manually. Each trading partner profile is identified by a unique ID determined by the enterprise, and delivery channels can be configured for acknowledgments, compression, industry-standard encryption and decryption, and nonrepudiation.

At run time, all steps in the business process, from initial receipt of the message to final delivery to the trading partner, are tracked in real time and also stored in the eXchange database. The Web-based message/package tracker provides tools for retrieving and filtering tracked message and envelope information. Used in conjunction with the other monitoring tools of the ICAN suite, this provides the enterprise with a complete solution for troubleshooting and managing all eBusiness activities.

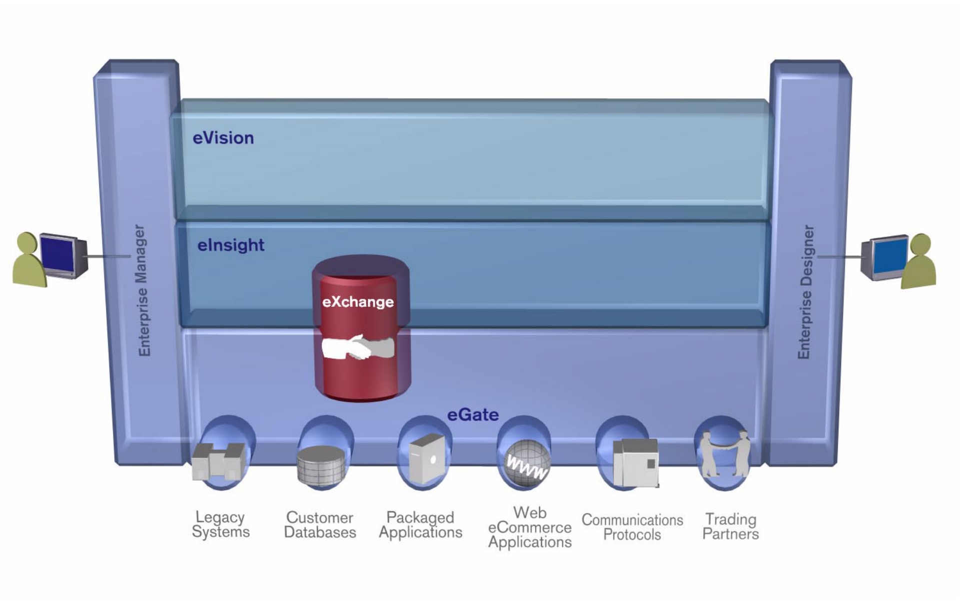
2.2 eXchange and the ICAN Suite

eXchange is part of the SeeBeyond ICAN Suite of products. eXchange provides a Web-based trading partner configuration and management solution for automating and securely managing business partner relationships for real-time interaction between the enterprise and its partners, suppliers, and customers.

2.2.1 ICAN Integration

eXchange is tightly integrated with the ICAN Suite and runs as a component within the ICAN Suite environment. Figure 1 illustrates how eXchange and other ICAN Suite components work together.

Figure 1 eXchange and the ICAN Suite



2.3 Architectural Overview

eXchange centers around the concept of a *Delivery Channel Profile* for each trading partner relationship. Delivery Channel Profiles are configured within eXchange for use by runtime components. Each profile specifies which B2B protocol(s) to use, where and how to receive inbound messages from trading partners, how to configure and secure messages in this channel, and how and where to deliver outbound messages to trading partners.

eXchange uses the following key components:

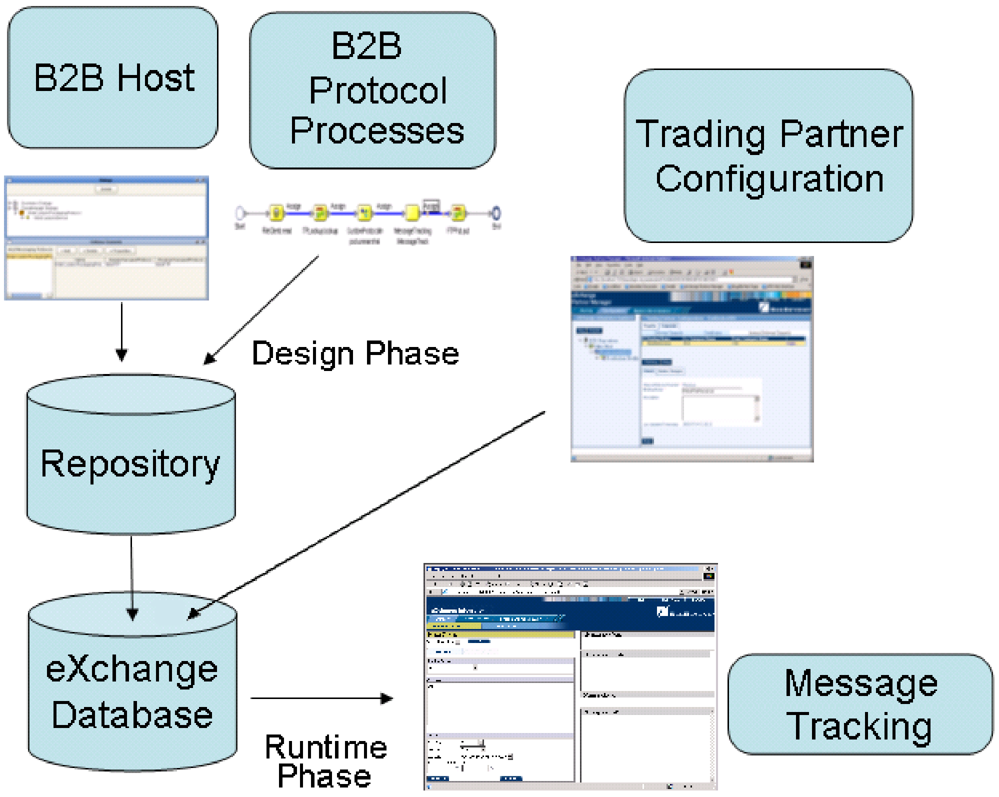
- **B2B Host Designer** — Using the **Enterprise Designer** GUI framework, eXchange provides an editor for setting up B2B environments, called the *B2B Host Designer*. Each B2B Host provides one or more protocol-specific delivery channels that are exposed to the eXchange database via the Repository. Delivery channels provided by the B2B Host can then be accessed by specific trading partners and reused. See **Figure 5 on page 22**.
- **B2B Services and Protocols** — eXchange provides two other special editors: The *eXchange Service Designer*, for modeling the choreography of B2B interactions between the internal system and an external trading partner, as mediated by the B2B Host; and the *eXchange Protocol Designer*, for setting up the message flow logic (*B2B protocol processes*) required to address a specific business challenge, using such activity elements as branching activities, timers, and exception handling. See **Figure 4 on page 21**.

  Prebuilt B2B protocol processes for such industry-standard B2B protocols as AS2 and ebXML are available as separately installable add-on products. eXchange also provides the flexibility of allowing the enterprise to create and configure custom protocol processes. See **Figure 4 on page 21**.

  eXchange also supplies **Channel Manager**, a collection of eXchange Services that provides trading partner–specific integration to the enterprise. Industry-standard transport protocols (FTP, HTTP, HTTPS, SMTP) are supported by Channel Manager and other eWays.
- **Trading Partner Configuration** — eXchange provides a Web-based GUI, eXchange Partner Manager (ePM), for configuring and managing B2B trading partners. Each trading partner has one or more delivery channels that specify the protocols to be used, with corresponding transport mechanisms—encryption parameters such as certificate, signature, and keystore information, acknowledgment-handling preferences, and so forth. See **Figure 6 on page 22**.
- **eXchange Database** — eXchange uses an Oracle database to mediate retrieval of trading partner information and to store run-time information on message tracking.
- **Message Tracking**— eXchange provides a specific MessageTracker application that can be combined with other processes in a project just by dragging it into the Connectivity Map, as well as a Web-based message tracking GUI with powerful filtering and searching capabilities. See **Figure 7 on page 23**.

The interaction of these components is illustrated in Figure 2.

Figure 2 eXchange Architecture



The illustrations in Figure 3, Figure 4, and Figure 5 indicate some of the features provided by the various GUIs.

Figure 3 User-Created B2B Protocol Process

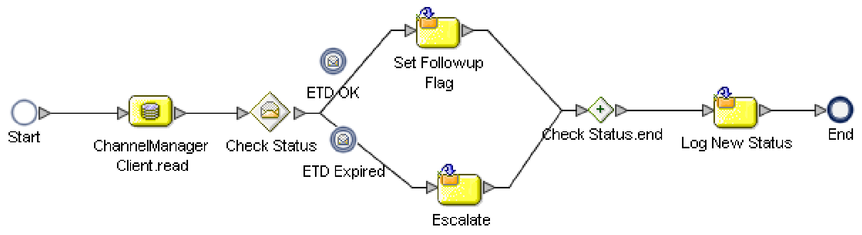


Figure 4 Prebuilt B2B Protocol Process (for AS2 Inbound)

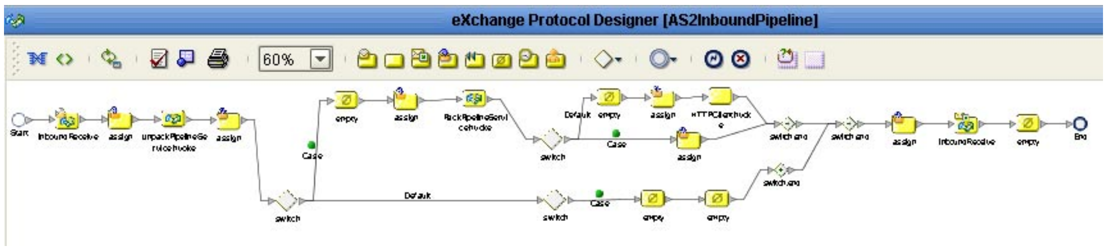
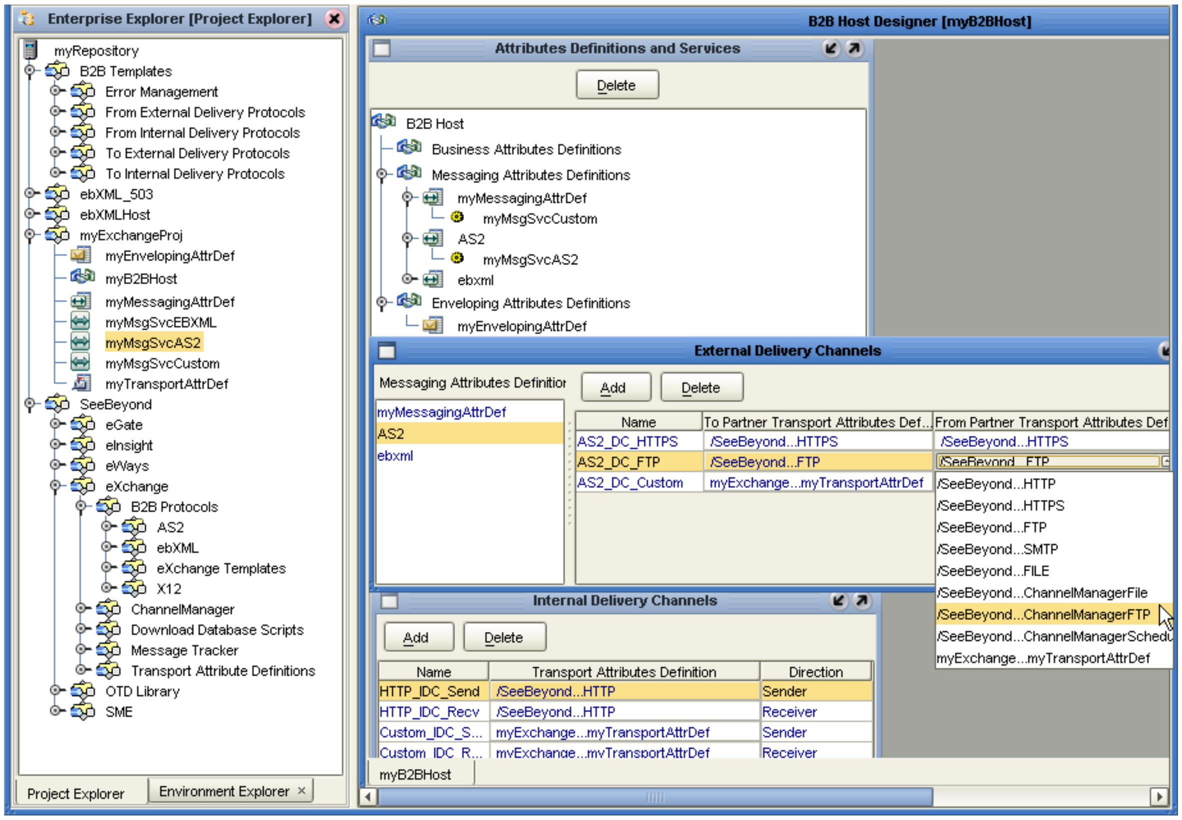


Figure 5 B2B Host Designer in Enterprise Designer

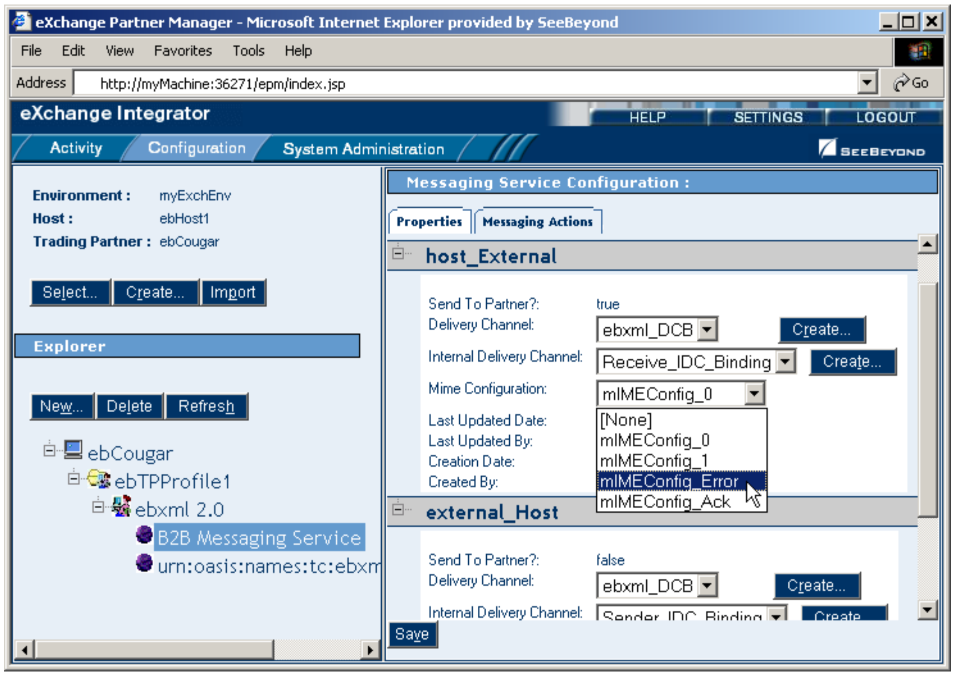


Name	To Partner Transport Attributes Def...	From Partner Transport Attributes Def
AS2_DC_HTTPS	/SeeBeyond...HTTPS	/SeeBeyond...HTTPS
AS2_DC_FTP	/SeeBeyond...FTP	/SeeBeyond...FTP
AS2_DC_Custom	myExchange...myTransportAttrDef	

/SeeBeyond...HTTP
/SeeBeyond...HTTPS
/SeeBeyond...FTP
/SeeBeyond...SMTP
/SeeBeyond...FILE
/SeeBeyond...ChannelManagerFile
/SeeBeyond...ChannelManagerFTP
/SeeBeyond...ChannelManagerSchedu
myExchange...myTransportAttrDef

Name	Transport Attributes Definition	Direction
HTTP_IDC_Send	/SeeBeyond...HTTP	Sender
HTTP_IDC_Recv	/SeeBeyond...HTTP	Receiver
Custom_IDC_S...	myExchange...myTransportAttrDef	Sender
Custom_IDC_R...	myExchange...myTransportAttrDef	Receiver

Figure 6 eXchange Trading Partner Configuration



Address: http://myMachine:36271/epm/index.jsp

Environment : myExchEnv
Host : ebHost1
Trading Partner : ebCougar

host_External

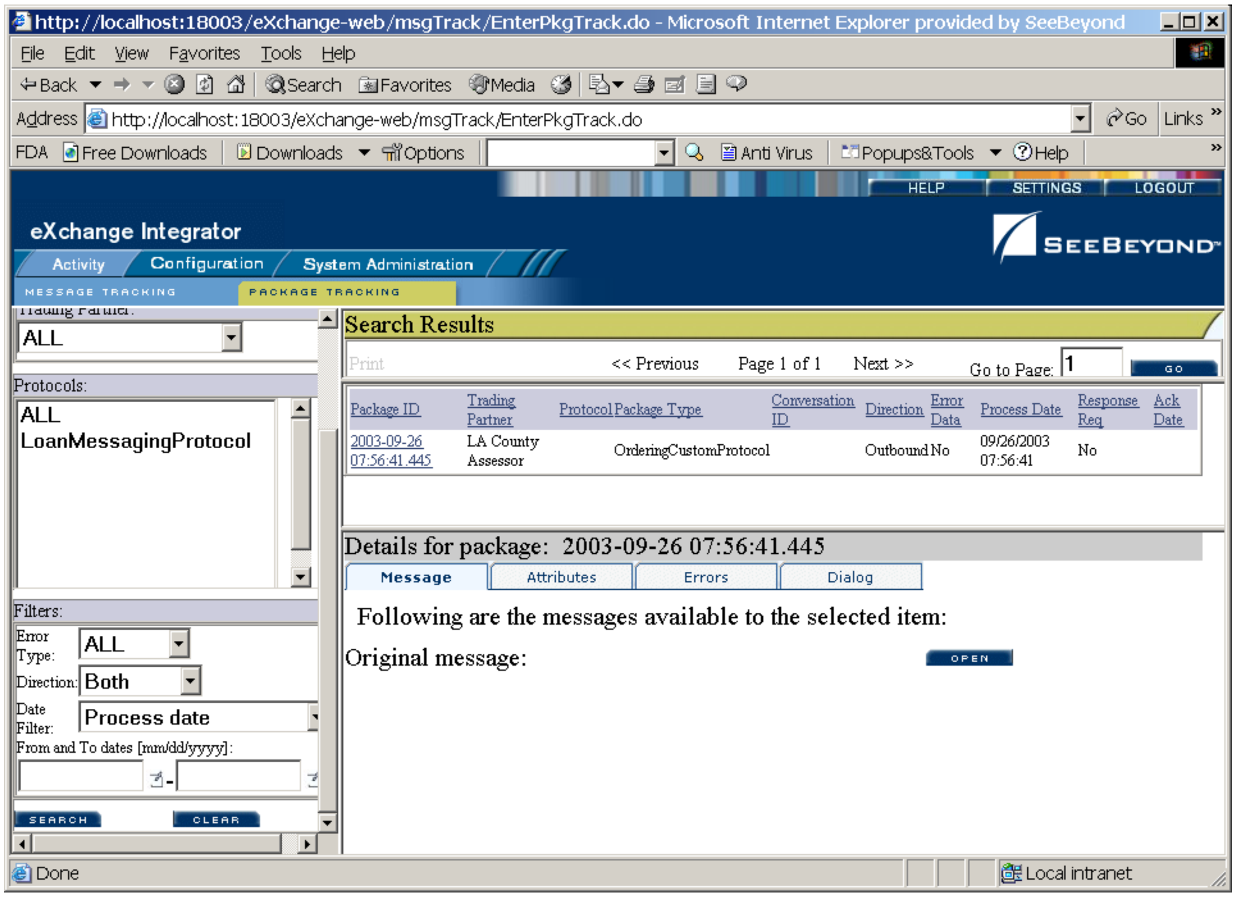
Send To Partner?: true
Delivery Channel: ebxml_DCB
Internal Delivery Channel: Receive_IDC_Binding
Mime Configuration: mIMEConfig_0
Last Updated Date:
Last Updated By:
Creation Date:
Created By:

[None]
mIMEConfig_0
mIMEConfig_1
mIMEConfig_Error
mIMEConfig_Ack

external_Host

Send To Partner?: false
Delivery Channel: ebxml_DCB
Internal Delivery Channel: Sender_IDC_Binding

Figure 7 eXchange Message Tracking


http://localhost:18003/eXchange-web/msgTrack/EnterPkgTrack.do - Microsoft Internet Explorer provided by SeeBeyond
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:18003/eXchange-web/msgTrack/EnterPkgTrack.do
Go Links
FDA Free Downloads Downloads Options Anti Virus Popups&Tools Help
HELP SETTINGS LOGOUT
eXchange Integrator
SEEBEYOND
Activity Configuration System Administration
MESSAGE TRACKING PACKAGE TRACKING
SEARCH CLEAR
OPEN
GO
Done
Local intranet

Trading Partner: ALL

Protocols: ALL, LoanMessagingProtocol

Filters:
Error Type: ALL
Direction: Both
Date Filter: Process date
From and To dates [mm/dd/yyyy]:

Search Results

Print << Previous Page 1 of 1 Next >> Go to Page: 1

Package ID	Trading Partner	Protocol	Package Type	Conversation ID	Direction	Error Data	Process Date	Response Req	Ack Date
2003-09-26 07:56:41.445	LA County Assessor	OrderingCustomProtocol			Outbound	No	09/26/2003 07:56:41	No	

Details for package: 2003-09-26 07:56:41.445

Message | Attributes | Errors | Dialog

Following are the messages available to the selected item:

Original message:

2.4 Process Overview

Using eXchange to create a business solution consists of three phases:

- Design phase within Enterprise Designer
- Design phase within eXchange Partner Manager
- Runtime phase

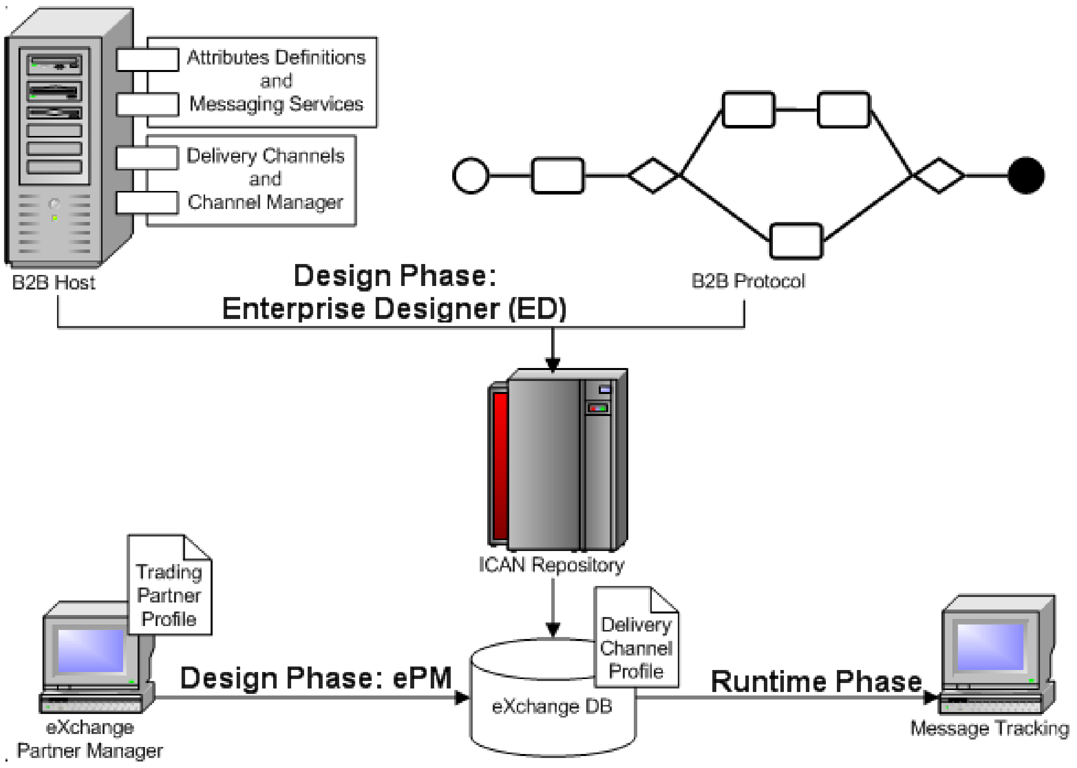
The purpose of the design phases is to: Create metadata for Delivery Channel Profiles; set up business logic for B2B protocol processes; configure connections with external systems; create and configure trading partners; and associate each trading partner relationship with a Delivery Channel Profile (DCP) configuration. Activating a trading partner exposes its DCP configuration settings to the eXchange database.

At run time, the Logical Host reads the DCP configuration from the database to determine: How to receive and process inbound messages; which business logic to run; and how to process and deliver outbound messages. Results are written to the database, where they can be filtered and viewed by the Message Tracker facility.

These phases are illustrated in Figure 8 and explained in further detail in the following sections:

- **Design Phase: Using Enterprise Designer** on page 24
- **Design Phase: Using eXchange Partner Manager** on page 25
- **Runtime Phase** on page 25

Figure 8 eXchange Architecture



2.4.1 Design Phase: Using Enterprise Designer

Within Enterprise Designer, the B2B Host Designer is used to create B2B Hosts. Each B2B Host is a logical collection of:

- *Messaging services* associated with either standard pre-built B2B protocols (such as AS2 or ebXML) or custom B2B protocols.
- *Attribute definitions* for transport, enveloping (packaging), and messaging.

The B2B Host Designer is used to choose which messaging services are to be used, and to associate each service with one or more sets of transport attribute definitions (such as HTTP, FTP, and many others) to create the metadata for a Delivery Channel Profile—in other words, the *types* of parameters to be supplied for transporting, [de]enveloping, and [receiving]sending messages [from]to trading partners.

After the B2B Host is set up, a Connectivity Map is created to connect its output, and the output of the Message Tracker application, to an Oracle database. Activation causes the DCPs to be stored in the database, and also creates an eXchange Service as an external server in the same Environment that contains the Oracle external. As needed, the eXchange Service corresponding to the B2B Host is configured with keystores, trust stores, and certificates for authentication and nonrepudiation.

Standard B2B Protocol processes (such as the add-ons for AS2 or ebXML) can be used, and/or the eXchange Protocol Designer can be used to create and configure business logic in custom B2B Protocol processes in the same way the eInsight Business Process Designer is used for Business Processes.

B2B Protocol processes for inbound and/or outbound messages are dragged into a Connectivity Map, where they are represented as services. There, they are connected in the usual fashion with externals (including the eXchange Service for channel management) and with other services. Activation of a corresponding Deployment Profile exposes the map's components for processing by Logical Hosts. As before, it also stores the DCPs into the eXchange database, making them available to eXchange Partner Manager.

2.4.2 Design Phase: Using eXchange Partner Manager

eXchange Partner Manager (ePM) is used to create and configure trading partners and to create trading partner profiles—an association between a particular trading partner and a set of Delivery Channel Profile parameters. For example, if a DCP uses HTTP, then each trading partner profile must be supplied with a value for the URL parameter; or, if a DCP uses FTP, then each trading partner profile must be supplied with values for hostname, target directory, and so forth.

Activating a trading partner stores all of its profiles' DCP configuration settings into the eXchange database.

2.4.3 Runtime Phase

The Logical Host reads the DCP configuration and receives inbound messages from all the channels it references. The DCP parameters for each channel dictate how to handle the inbound message (acknowledgment, decryption, de-enveloping, authentication, ...); the business logic of the associated B2B Protocol and Connectivity Map provide further routing and processing; and for an outbound message, the DCP parameters dictate how to handle it (compression, encryption, signature, enveloping, ...) and how and where to send it.

eXchange also provides a Message Tracking facility for searching, filtering, and viewing all information written to the eXchange database—errors, acknowledgments, notifications, message attributes, and so forth.

Chapter 3

Installing eXchange

This chapter provides the prerequisites and steps for installing eXchange Integrator.

3.1 System Requirements

This section lists system requirements and database requirements. The *SeeBeyond ICAN Suite Installation Guide* and the **Readme.txt** file, available on the product media and via Enterprise Manager (Documentation tab), contain up-to-date operating system requirements for each supported platform.

3.1.1. Platform Support

eXchange supports the following operating systems:

- Microsoft Windows Server 2003, Windows XP SP1a, and Windows 2000 SP3 or SP4
- Sun Solaris 8 and 9, with required patches
- HP Tru64 V5.1A with required patches
- HP-UX 11.0 and 11i (PA-RISC), with required patches and parameter changes
- IBM AIX 5.1L and 5.2 (either 64-bit kernel or 32-bit kernel with 64-bit extension), with required maintenance level patches
- Red Hat Linux 8 (Intel *x*86) and Linux Advanced Server 2.1 (Intel *x*86)

3.1.2. Database Support

Database for eXchange Partner Management and Message Tracking

The eXchange database provides a run-time persistent store for trading partner management and message tracking. eXchange supports the following databases:

- Oracle 8.1.7
- Oracle 9.01
- Oracle 9.2

3.2 Installation Steps

The steps for installing eXchange are the same as for other products in the ICAN Suite. You can find general product installation instructions in the *ICAN Suite Installation Guide,* which is available on the product media and can also be accessed via Enterprise Manager (Documentation tab).

3.2.1. Uploading eXchange to the Repository

Before you begin

- A Repository server must be running on the machine where you will be uploading the eXchange product files.
- The following ICAN **.sar** files must have already been uploaded to this Repository:
  - eGate Enterprise Designer (**eGate.sar**) 5.0.4
  - Batch eWay adapter (**BatcheWay.sar**) 5.0.4
  - Oracle eWay adapter (**OracleeWay.sar**) 5.0.4
  - File eWay adapter (**FileeWay.sar**) — This not an installation requirement, but it is required by the sample implementation

  Additionally, if you will be using either AS2 or ebXML, the following must also be uploaded to this Repository:
  - HTTP(S) eWay adapter (**HTTPeWay.sar**) 5.0.4
  - Secure Message Extension (**SMEWebServices.sar**) 5.0.4

Note: *SMEWebServices.sar is required for such features as encryption/decryption, signature verification, certificate authentication, and compression/decompression.*

To upload eXchange product files to the Repository

1. On a Windows machine, start a Web browser and point it at the machine and port where the Repository server is running:

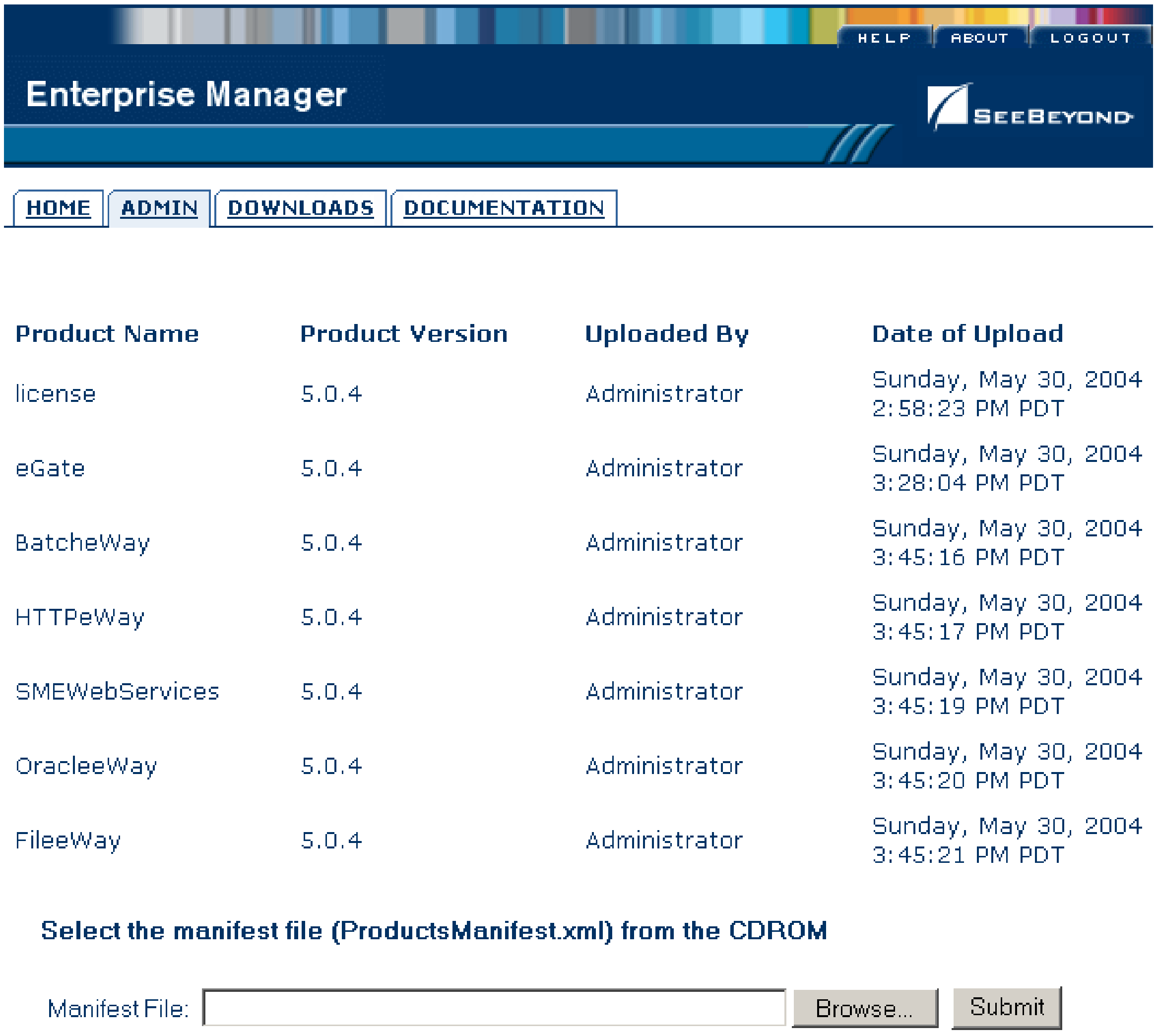
   ```
   http://<hostname>:<port>
   ```

   where
   - *<hostname>* is the name of the machine running the Repository server.
   - *<port>* is the starting port number assigned when the Repository was installed.

   For example, the URL you enter might look like either of the following:

   ```
   http://localhost:12001
   http://serv1234.company.com:19876
   ```
2. On the Enterprise Manager **SeeBeyond Customer Login** page, enter your username and password.
3. When Enterprise Manager responds, click the **ADMIN** tab. See Figure 9.

Figure 9 Enterprise Manager ADMIN page



Product Name	Product Version	Uploaded By	Date of Upload
license	5.0.4	Administrator	Sunday, May 30, 2004 2:58:23 PM PDT
eGate	5.0.4	Administrator	Sunday, May 30, 2004 3:28:04 PM PDT
BatcheWay	5.0.4	Administrator	Sunday, May 30, 2004 3:45:16 PM PDT
HTTPeWay	5.0.4	Administrator	Sunday, May 30, 2004 3:45:17 PM PDT
SMEWebServices	5.0.4	Administrator	Sunday, May 30, 2004 3:45:19 PM PDT
OracleeWay	5.0.4	Administrator	Sunday, May 30, 2004 3:45:20 PM PDT
FileeWay	5.0.4	Administrator	Sunday, May 30, 2004 3:45:21 PM PDT

Select the manifest file (ProductsManifest.xml) from the CDROM

4 In the ADMIN page, click **Browse**.

5 In the **Choose file** dialog, click **ProductsManifest.xml**, and then click **Open**.

6 In the ADMIN page, click **Submit**.

The lower half of the ADMIN page lists the product files you are licensed to upload.

7 In the Products column, find **eXchange**, and then click the **Browse** button for it.

8 In the **Choose file** dialog, click **eXchange.sar**, and then click **Open**.

9 Repeat the previous two steps for other eXchange-related product **.sar** files you are licensed to upload, such as SME Web Services (for **S**ecure **M**essaging **E**xtension), or one or more of the standard protocols (such as for AS2 or ebXML) or OTD libraries (such as for X12, HIPAA, or UN/EDIFACT).

10 For documentation and samples, also upload the corresponding [...]**Docs.sar** files, such as eXchangeDocs.sar, SMEWebServicesDocs.sar, and so forth.

11 In the ADMIN page, click the | upload now :: | button.

3.2.2. Refreshing Enterprise Designer with eXchange

Before you begin

- You must have already downloaded and installed Enterprise Designer.
- A Repository server must be running on the machine where you uploaded the eXchange product files.

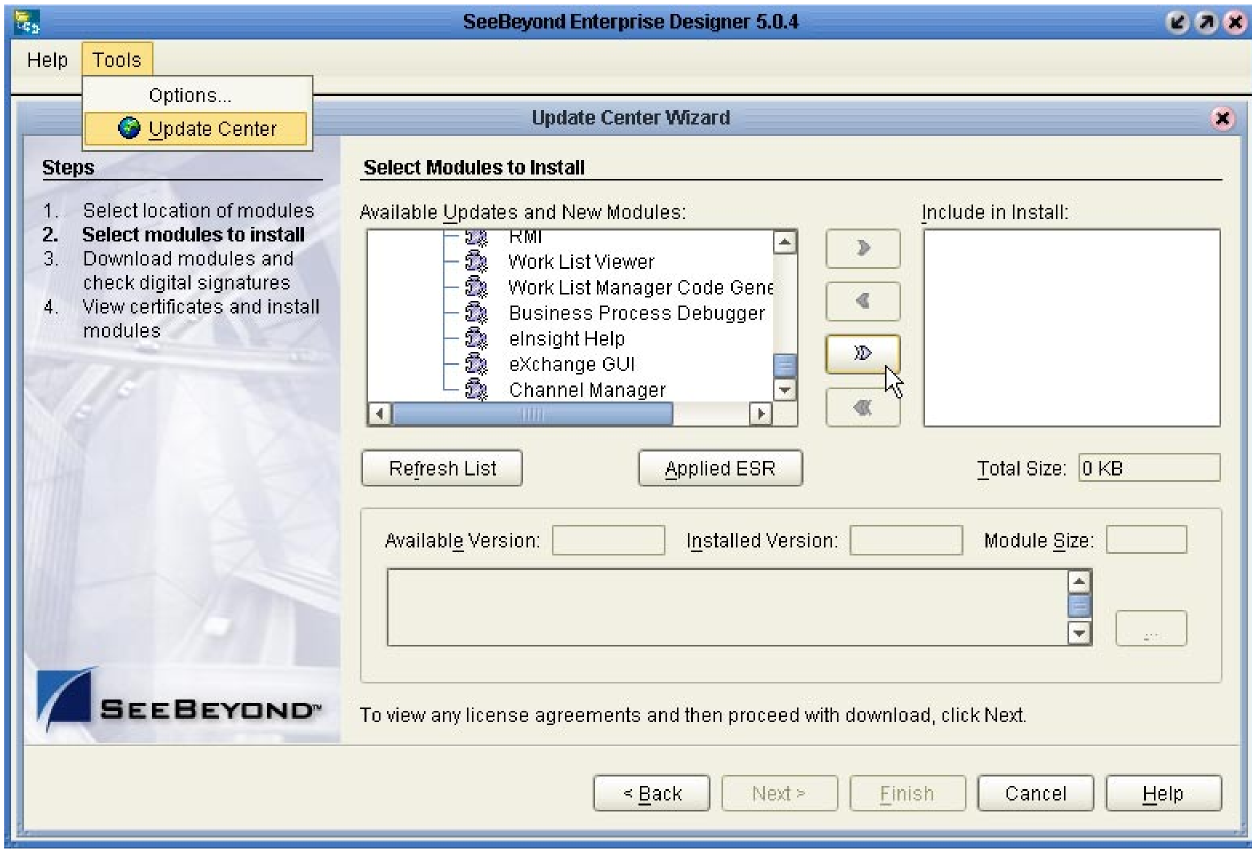
To refresh an existing installation of Enterprise Designer

1 Start Enterprise Designer.

2 On the **Tools** menu, click **Update Center**.

The Update Center shows a list of components ready for updating. See Figure 10.

Figure 10 Update Center Wizard: Select Modules to Install


SeeBeyond Enterprise Designer 5.0.4
Help
Tools
Options...
Update Center
Update Center Wizard
Steps
1. Select location of modules
2. Select modules to install
3. Download modules and check digital signatures
4. View certificates and install modules
Select Modules to Install
Available Updates and New Modules:
RMI
Work List Viewer
Work List Manager Code Gene
Business Process Debugger
eInsight Help
eXchange GUI
Channel Manager
Include in Install:
Refresh List
Applied ESR
Total Size: 0 KB
Available Version:
Installed Version:
Module Size:
To view any license agreements and then proceed with download, click Next.
< Back
Next >
Finish
Cancel
Help

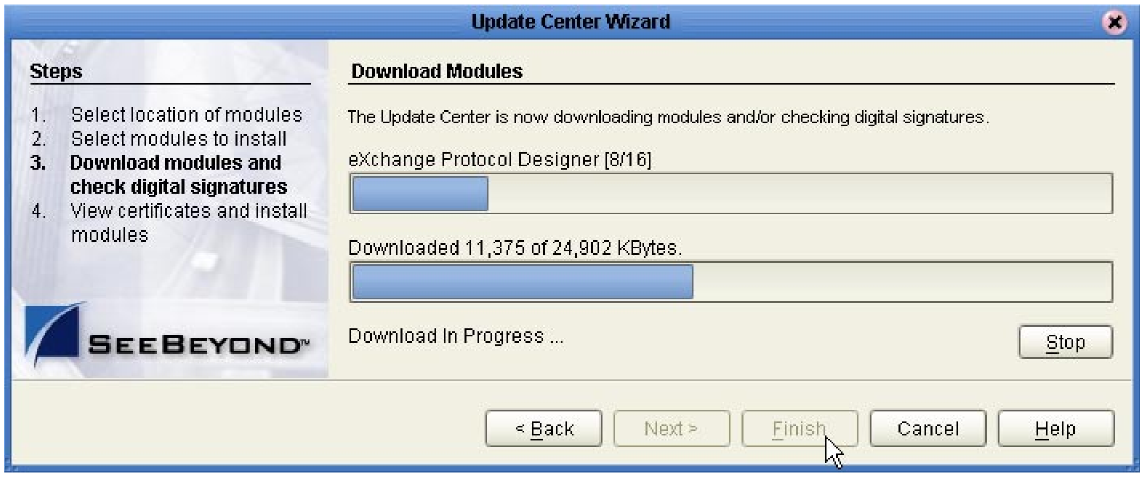
3 Click **Add All** (the button with a doubled chevron pointing to the right).

All modules move from the Available/New pane to the **Include in Install** pane.

4 Click **Next** and, in the next window, click **Accept** to accept the license agreement.

The wizard shows you the progress of the download. See Figure 11.

Figure 11 Update Center Wizard: Progress Bars


Update Center Wizard
Steps
1. Select location of modules
2. Select modules to install
3. Download modules and check digital signatures
4. View certificates and install modules
SEEBEYOND
Download Modules
Stop
< Back
Next >
Finish
Cancel
Help

The Update Center is now downloading modules and/or checking digital signatures.

eXchange Protocol Designer [8/16]

Downloaded 11,375 of 24,902 KBytes.

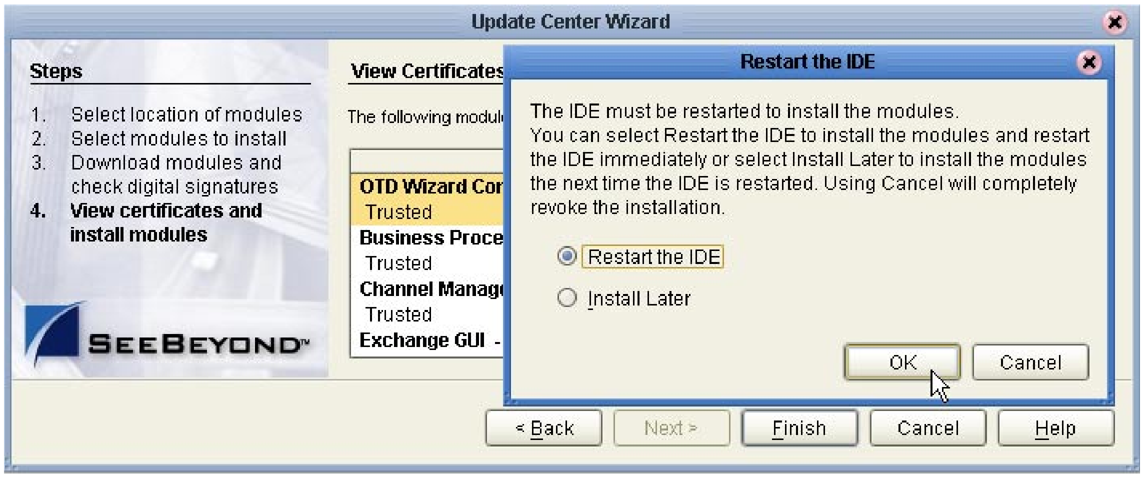
Download In Progress ...

5 When the progress bars indicate the download has ended, click **Next**.

6 Review the certificates and installed modules, and then click **Finish**.

7 When prompted to restart Enterprise Designer, click **OK**. See Figure 12.

Figure 12 Update Center Wizard: Restart Enterprise Designer



The IDE must be restarted to install the modules.
You can select Restart the IDE to install the modules and restart the IDE immediately or select Install Later to install the modules the next time the IDE is restarted. Using Cancel will completely revoke the installation.

[✓ Restart the IDE]
Install Later

When Enterprise Designer restarts, the installation of eXchange Integrator is complete, and you can use all eXchange tools provided on the Enterprise Designer framework.

3.3 Database Scripts

eXchange provides database scripts to create the database schemas for eXchange. (The eXchange database schema collects and persists data about your trading partner profiles, and also allows you to track message delivery history.)

For eXchange, the areas to be configured are:

- **Creating and Configuring the eXchange Database Instance** on page 31
- **Extracting and Customizing Database Scripts for eXchange** on page 32
- **Running Database Scripts to Set Up the eXchange Database** on page 33

3.3.1. Creating and Configuring the eXchange Database Instance

Before you begin: You need to have already created an Oracle database instance with an entry in the **tnsnames.ora** file. Your TNSlistener service must be running, and you will need to know the name of the database instance (default: **eXchange**) and to temporarily use the system username and password (default: **sys,manager**).

If you have never installed an Oracle database, ask your Oracle database administrator for help. The following constitutes a brief reminder of how to use the Oracle 9i wizard.

To create a new Oracle database instance

1. Step 1 ("Operations"): Choose **Create a database**.
2. Step 2 ("Database Templates"): Choose **New Database**.
3. Step 3 ("Database Identification"): Enter (for example) **eXchange**
4. Step 4 ("Database Features"): Deselect all checkboxes and reply **Yes** to all prompts.
5. Step 5 ("Database Connection Options"): Choose **Dedicated** [...].
6. Step 6 ("Initialization Parameters"): Keep all values unchanged.
7. Step 7 ("Database Storage"): Under Datafiles, click \{DB_Name}**undotbs01.dbf** (the fifth entry). In the **General** tab, reduce **File Size** from 200 to 100.
8. Step 8 ("Creation Options"): Choose **Create Database**, and then click Finish.

Modifying the init.ora File for the eXchange Database

If you create a new database, you must increase the **open_cursors** parameter for the eXchange database to 500. When you open this file, verify that it contains the line specified in step 3 below, and add it if it is not in the file.

To modify the value of open_cursors in the init.ora file

1. Navigate to <*Oracle home*>**\admin\**<*eXchange database name*>**\pfile\**. For example:

   ```
   cd C:\oracle\admin\exchange\pfile
   ```

2. Use a text editor to open the **init.ora** file in this folder. For example:

   ```
   notepad init.ora
   ```

3. Search for the text **open_cursors** and change its setting to 500, as follows

   ```
   open_cursors = 500
   ```

4. Save the file.
5. Restart the database.

Next: To set up the eXchange database, you will extract the files supplied in the Database Scripts project folder, edit scripts so they have the correct parameters for your Oracle database setup, and run the scripts to set up and initialize the database.

3.3.2. Extracting and Customizing Database Scripts for eXchange

Note: *Do not skip this section. At this release, even if you already have a pre-existing eXchange database, you will need to extract and eventually run the* ***createdb*** *script.*

To extract and customize the database scripts

1. In Enterprise Explorer, in the project tree, expand the following folders: **SeeBeyond > eXchange** > **Download Database Scripts**
2. Right-click **oracle.zip** and, on the popup context menu, click **Export**; then use the **Save** dialog box to save the file to a local directory.
3. Extract the files in **oracle.zip** into this local directory, yielding:
   - createdb.cmd
   - createdb.sql
   - createtablespaces.cmd
   - createtablespaces.sql
   - createuser.sql
   - eXchange50Runtime.sql
   - in_user_seq.sql
   - setenv.cmd
   - x12_datamodel.sql
4. If your Oracle location is not **c:\oracle\oradata**, or if your database instance name (SID) is other then **eXchange**, then open the **createtablespaces.sql** file and make the appropriate change or changes in the first line.

Next: Further steps are required; however, the next steps you take depend on whether you are creating a new database or upgrading an existing one:

- If you have an eXchange database from 5.0.3, skip ahead to **"Updating an eXchange 5.0.3 Database to 5.0.4" on page 35**.
- If you have an eXchange database from 5.0.1 or 5.0.2, skip ahead to **"Updating an eXchange 5.0.1 or 5.0.2 Database to 5.0.3" on page 35**.
- If you have an eXchange database from 5.0, skip ahead to **"Updating an eXchange 5.0 Database" on page 34**.
- If you do not already have an eXchange database, continue with **"Running Database Scripts to Set Up the eXchange Database" on page 33**.

Important: *The database user who runs the* ***.sql*** *scripts must have permission to create tables.*

3.3.3. Running Database Scripts to Set Up the eXchange Database

Note: *If you already created and populated an eXchange database for 5.0 or later, skip ahead to* **"Updating an eXchange 5.0 Database" on page 34** *or later.*

To run the database scripts that install the schema

1. Open a command prompt, change directories to the local directory where you saved the **.sql** scripts in the previous procedure, and enter the following SQL*Plus command:

   ```
   <path>\sqlplus system/<SYSTEMPWD>@<TNSNAME> @createtablespaces.sql
   ```

   where:

   <SYSTEMPWD> is the password for the **system** login ID

   <TNSNAME> is the name of the Oracle database instance you created for eXchange.

   Here are two examples of valid commands, depending on the password and name:

   ```
   C:\oracle\ora92\bin\sqlplus system/manager1@eX50 @createtablespaces.sql
   sqlplus system/oraclePW@eXchange @createtablespaces.sql
   ```

   When this finishes, you will have created new tablespaces.

2. In the command prompt, enter the following SQL*Plus command:

   ```
   sqlplus system/<SYSTEMPWD>@<TNSNAME> @createuser.sql
   ```

   where, as before, *<SYSTEMPWD>* is the password for the **system** login ID and *<TNSNAME>* is the name of the Oracle database instance you created for eXchange.

   Here is an example of a valid command:

   ```
   \oracle\ora92\bin\sqlplus system/myPassWd@eX504DB @createuser.sql
   ```

3. In response to the system prompt for value #1, enter the username. For example: **ex_admin**

4. In response to the system prompt for value #2, enter the password. For example: **ex_admin**

5. After running the previous two SQL scripts, there is one more. In the command prompt, enter the following SQL*Plus command:

   ```
   sqlplus ex_admin/ex_admin@<TNSNAME> @createdb.sql
   ```

   where, as before, *TNSNAME* is the name of the eXchange Oracle database instance, and your eXchange administrator username and password are both **ex_admin**.

Result: After the **createdb.sql** script ends, you are done—you do not need to run any further SQL scripts. The system populates the tables, and you are ready to use the database instance as your eXchange database. You can create Oracle OTDs based on this database, and use Enterprise Manager GUIs for trading partner configuration and message tracking. If you will be using Secure Messaging Extension, see **"Additional Policy JAR Files Required to Run SME" on page 36**. If you will be using ebXML, see **"Additional JAR Files for ebXML Encryption" on page 37**.

Updating an eXchange 5.0 Database

Note: *The following steps should be performed only if you have a pre-existing eXchange database from 5.0. If, instead, you have a database from 5.0.1 or 5.0.2 or one that has been upgraded to 5.0.1, skip ahead to the procedure* **"Updating an eXchange 5.0.1 or 5.0.2 Database to 5.0.3" on page 35**. *If, instead, you have a database from 5.0.3, or one that has been upgraded to 5.0.3, skip ahead to the procedure* **"Updating an eXchange 5.0.3 Database to 5.0.4" on page 35**.

To extract and customize the database script

1. In Enterprise Explorer, in the project tree, expand the following folders: **SeeBeyond** > **eXchange** > **Download Database Scripts**
2. Right-click **oracle_upgrade_500_to_501.zip** and, on the popup context menu, click **Export**; then use the **Save** dialog box to save the file to a local directory.
3. Extract the files in **oracle_upgrade_500_to_501.zip** into this local directory, yielding:
   - eXchange_500_to_501_script.sql
4. In a command prompt, change to the directory where you extracted these files, start a SQL*Plus session and log into the eXchange database instance. For example:

```
G:\> cd \ican50\dbscripts
G:\ican50\dbscripts> C:\oracle\ora92\bin\sqlplus
SQL*Plus: Release 9.2.0.1.0 - Production on Mon Jun 21 13:38:26 2004
Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.
Enter user-name: ex_admin
Enter password:
Connected to:
Oracle9i Enterprise Edition Release 9.2.0.1.0 - Production
With the Partitioning, OLAP and Oracle Data Mining options
JServer Release 9.2.0.1.0 - Production
SQL>
```

5. Execute the **eXchange_500_2_501_script.sql** script, wait for the prompt to reappear, and then exit. The console should resemble the following:

```
SQL> @eXchange_500_2_501_script.sql
Table altered.
[...]
Table created.
Input truncated to 1 characters
Procedure created.
PL/SQL procedure successfully completed.
2 rows deleted.
Commit complete.
Input truncated to 1 characters
Procedure created.
PL/SQL procedure successfully completed.
Table dropped.
Commit complete.
[...]
Index created.
SQL> exit
```

Next: Further steps are required; now that you have upgraded your database to 5.0.1 (same as 5.0.2), continue with the following procedure, **"Updating an eXchange 5.0.1 or 5.0.2 Database to 5.0.3" on page 35**.

Updating an eXchange 5.0.1 or 5.0.2 Database to 5.0.3

Note: *The following steps should be performed only if you have a pre-existing eXchange database from 5.0.1 or 5.0.2, or one that has been upgraded to 5.0.1. If, instead, you have a database from 5.0.3, or one that has been upgraded to 5.0.3, skip ahead to the procedure* **"Updating an eXchange 5.0.3 Database to 5.0.4" on page 35**.

To extract, customize, and run the database script

1. In Enterprise Explorer, in the project tree, expand the following folders: **SeeBeyond > eXchange > Download Database Scripts**
2. Right-click **oracle_upgrade_502_to_503.zip** and, on the popup context menu, click **Export**; then use the **Save** dialog box to save the file to a local directory.
3. Extract the files in **oracle_upgrade_502_to_503.zip** to this local directory, yielding:
   - eXchange_502_to_503_script.sql
4. In a command prompt, change to the directory where you extracted these files, start a SQL*Plus session and log into the eXchange database instance. For example:

```
G:\> cd \ican50\dbscripts
G:\ican50\dbscripts> C:\oracle\ora92\bin\sqlplus
SQL*Plus: Release 9.2.0.1.0 - Production on Mon Jun 21 13:41:18 2004
Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.
Enter user-name: ex_admin
Enter password:
[...]
SQL>
```

5. Execute the **eXchange_502_2_503_script.sql** script, wait for the prompt to reappear, and then exit. The console should resemble the following:

```
SQL> @eXchange_502_2_503_script.sql
Table altered.
Table altered.
[...]
Commit complete.
Table created.
Index created.
SQL> exit
```

Updating an eXchange 5.0.3 Database to 5.0.4

Note: *The following steps should be performed only if you have a pre-existing eXchange database from 5.0.3, or one that has been upgraded to 5.0.3.*

To extract and customize the database script

- Open a command prompt, change directories to the location where you extracted the **.sql** scripts from **oracle.zip** (see **"Extracting and Customizing Database Scripts for eXchange" on page 32**), and enter the following SQL*Plus command:

```
sqlplus ex_admin/ex_admin@<TNSNAME> @createdb.sql
```

  where, as before, *`TNSNAME`* is the name of the eXchange Oracle database instance, and your eXchange administrator username and password are both assumed to be **ex_admin**.

After the **createdb.sql** script ends, you are done—you do not need to run any further SQL scripts, and you are ready to use the database instance as your eXchange database. You can create Oracle OTDs based on this database, and use Enterprise Manager GUIs for trading partner configuration and message tracking. If you will be using Secure Messaging Extension, see **"Additional Policy JAR Files Required to Run SME" on page 36**. If you will be using ebXML, see **"Additional JAR Files for ebXML Encryption" on page 37**.

3.4 Additional Policy JAR Files Required to Run SME

If you will be using encryption/decryption, and/or signatures and verification, and/or the compression/decompression libraries supplied with Secure Messaging Extension (SME), you must download and apply additional policy **.jar** files. The type of **.jar** files required depends on the JVM are using. Refer to your JVM vendor for exact details on the specific policy **.jar** file requirements.

Use Table 2 to determine which JRE is included in the eGate logical host.

Table 2 JRE Versions Listed by Operating System

Operating System	JRE	URL
Windows, Solaris, HP-UX, Linux, Tru64	1.4.2	**http://java.sun.com/j2se/1.4.2/download.html**
AIX	1.4.1	**http://java.sun.com/products/archive/j2se/1.4.1_07/index.html**

To download the required policy.jar files

1. Scroll to the bottom of the web page listed in Table 2 for your logical host's JRE.
2. Click the DOWNLOAD link for **Unlimited Strength Jurisdiction Policy Files 1.4.2**. (or, for AIX, **Unlimited Strength Jurisdiction Policy Files 1.4.1**).
3. Click the link to download the **.zip** file containing the required policy **.jar** files (in other words, for JRE 1.4.2, the link Download **jce_policy-1_4_2.zip**; or, for JRE 1.4.1, the link Download **jce_policy-1_4_1.zip**)
4. Extract the following required policy **.jar** files:
   - **local_policy.jar**
   - **US_export_policy.jar**
5. Then, for each of your logical hosts, replace the versions of these files in:

   ```
   <logicalhost>\jre\lib\security\
   ```
6. In addition, if you are running a repository on AIX, also replace the versions of these files in:

   ```
   <AIXrepository>/jre/1.4.x/security/
   ```

For complete information on SME, see the *Secure Messaging Extension User's Guide*.

3.5 Additional JAR Files for ebXML Encryption

If you will be using ebXML, you must download and apply an additional **.jar** file, xss4j.jar, for XML security. This can be downloaded, as part of the XML Security Suite, from the following Web site:

http://www.alphaworks.ibm.com/tech/xmlsecuritysuite

The only file needed is **xss4j.jar**. After extracting it, copy it to the following location for each logical host:

```
<logicalhost>\stcis\lib\
```

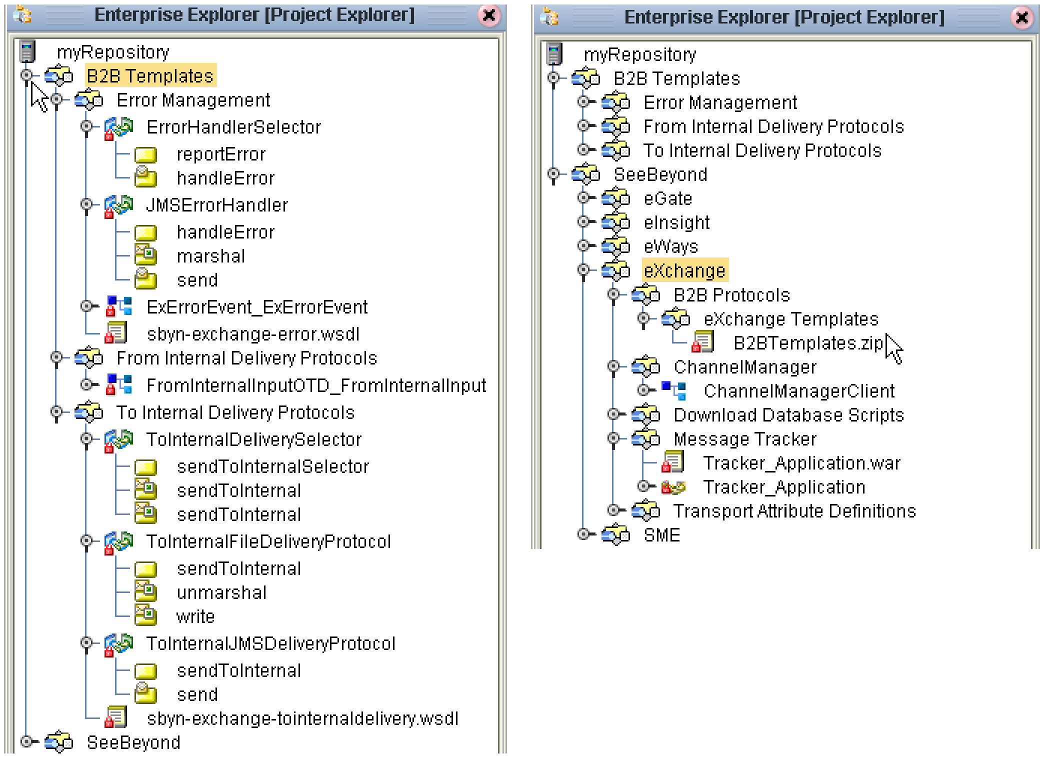
Chapter 4

eXchange Features

This chapter provides brief descriptions of components prepackaged with eXchange, as well as an overview of SME processes.

The components in the root **B2B Templates** folder are intended to be customized, and so the descriptions that appear here reflect the as-shipped versions of the components. Components in the SeeBeyond > **eXchange** folder must not be changed or customized. However, the SeeBeyond > eXchange > B2B Protocols > **eXchange Templates** folder contains the as-shipped versions of all *prepackaged* B2B protocols and B2B Templates; you can export any of these **.zip** files and then re-import it as a project. See Figure 13.

Figure 13 Prepackaged Folders: B2B Templates, and SeeBeyond > eXchange


Enterprise Explorer [Project Explorer]
Enterprise Explorer [Project Explorer]

```
myRepository
  B2B Templates
    Error Management
      ErrorHandlerSelector
        reportError
        handleError
      JMSErrorHandler
        handleError
        marshal
        send
      ExErrorEvent_ExErrorEvent
      sbyn-exchange-error.wsdl
    From Internal Delivery Protocols
      FromInternalInputOTD_FromInternalInput
    To Internal Delivery Protocols
      ToInternalDeliverySelector
        sendToInternalSelector
        sendToInternal
        sendToInternal
      ToInternalFileDeliveryProtocol
        sendToInternal
        unmarshal
        write
      ToInternalJMSDeliveryProtocol
        sendToInternal
        send
      sbyn-exchange-tointernaldelivery.wsdl
  SeeBeyond
```

```
myRepository
  B2B Templates
    Error Management
    From Internal Delivery Protocols
    To Internal Delivery Protocols
  SeeBeyond
    eGate
    eInsight
    eWays
    eXchange
      B2B Protocols
        eXchange Templates
          B2BTemplates.zip
      ChannelManager
        ChannelManagerClient
      Download Database Scripts
      Message Tracker
        Tracker_Application.war
        Tracker_Application
      Transport Attribute Definitions
    SME
```

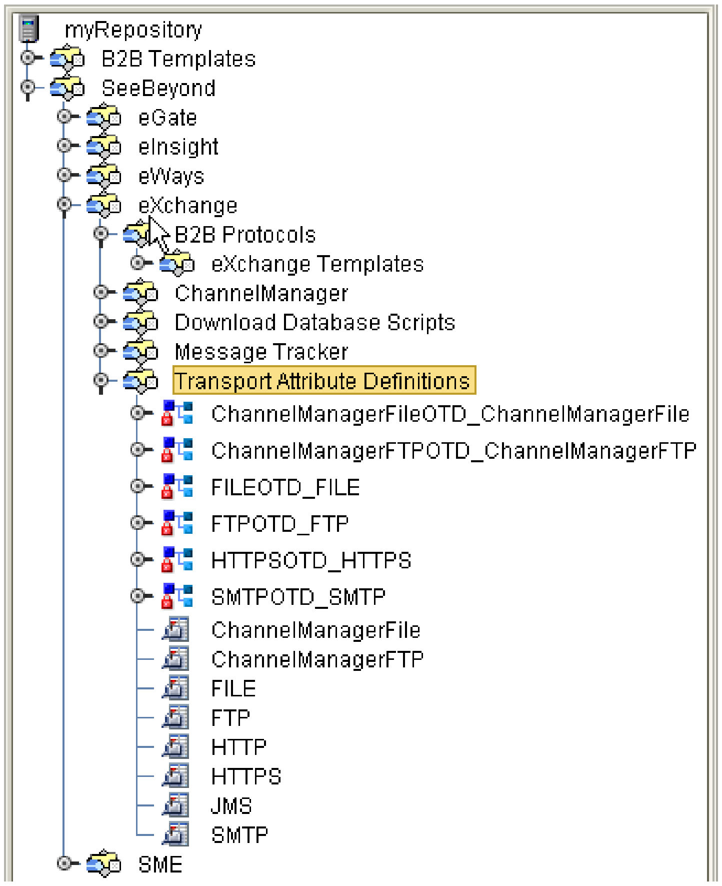
In this chapter

4.1 Transport Attribute Definitions

Transport attribute definitions provide the metadata required at the transport layer. (In this context, “metadata” means the *categories* of information, not any actual values.) Once a transport attributes definition has been included in a B2B host, it is exposed to ePM so that specific values can be supplied for specific trading partner configurations.

The SeeBeyond > eXchange > **Transport Attribute Definitions** folder (see Figure 14) contains eight transport attribute definitions (TADs) and corresponding OTDs.

Figure 14 SeeBeyond > eXchange > Transport Attribute Definitions Folder



Overview

Different transport protocols require different types of attributes; for example, HTTP requires little more than a URL, but FTP requires a username, password, hostname, port, path, and file pattern, and possibly other attributes as well. For this reason, the metadata for HTTP-based and FTP-based TADs are quite different. When a TAD is referenced by a delivery channel, its attributes govern the appearance and behavior of ePM for users who supply values for that channel.

At run time, a TAD’s metadata is made available to the application through the two methods of its associated OTD: **unmarshal** parses an inbound stream into an internal data structure, and **marshal** serializes the internal data into a linear outbound stream.

All TADs define their metadata using the format shown in Table 3.

Table 3 Metadata for All Transport Attribute Definitions

Field Name	Explanation
Name	The (internal) parameter name. Used programmatically; never seen in ePM.
Display	The parameter label as seen by the ePM user.
Type	Data type. Used programmatically; never seen in ePM.
Required	Checkbox governing whether a value must be supplied in ePM. If yes, ePM displays an red asterisk to signal the user that this is a required value.
Direction	FromPartner, ToPartner, or Both. Used programmatically.
Default	The value supplied before the ePM user takes action, or takes no action.
List of Values	Items to display in a drop-down list for the ePM user to choose from
Fixed	*(not used in any of SeeBeyond-supplied TADs)*
Format String	*(not used in any of SeeBeyond-supplied TADs)*

4.1.1. FILE

The File eWay uses the FILE transport attributes definition to read from a file or write to a file. When designating a pattern of files to be read, the * (asterisk) is a wildcard meaning “zero or more characters.”

Table 4 lists the attributes of the FILE TAD and corresponding OTD.

Table 4 Attributes for the FILE Transport Attributes Definition

Name	Display	Type	Req?	Direction	Default	List
FilePattern	FilePattern	String	Yes	Both		
Directory	Directory	String	Yes	Both		

4.1.2. FTP

For File Transfer Protocol, the BatchFTP eWay uses the FTP transport attributes definition to read from a file or write to a file in a remote location. When designating a pattern of files to be read, the * (asterisk) is a wildcard meaning “zero or more characters.”

Table 5 lists the attributes of the FTP TAD and corresponding OTD.

Table 5 Attributes for the FTP Transport Attributes Definition

Name	Display	Type	Req?	Direction	Default	List
FilePattern	FilePattern	String	Yes	Both		
Directory	Directory	String	Yes	Both		

Table 5 Attributes for the FTP Transport Attributes Definition (Continued)

Name	Display	Type	Req?	Direction	Default	List
UserName	UserName	String	Yes	Both		
Password	Password	Password	Yes	Both		
HostName	HostName	String	Yes	Both		
PortNumber	PortNumber	Integer	No	Both		
SocksEnabled	SocksEnabled	Boolean	No	Both	false	
SocksHostName	SocksHostName	String	No	Both		
SocksUserName	SocksUserName	String	No	Both		
SocksPassword	SocksPassword	Password	No	Both		
SocksServerPort	SocksServerPort	String	No	Both		

4.1.3. HTTP

For Hypertext Transfer Protocol, the HTTP(S) eWay can use the HTTP transport attributes definition to access Web pages. The HTTP TAD has no attributes and has no corresponding OTD.

4.1.4. HTTPS

For Hypertext Transfer Protocol over SSL (Secure Sockets Layer), the HTTP(S) eWay uses the HTTPS transport attributes definition to access secure Web page.

Table 6 lists the attributes of the HTTPS TAD and corresponding OTD.

Table 6 Attributes for the HTTPS Transport Attributes Definition

Name	Display	Type	Req?	Direction	Default	List
ClientCertAlias	ClientCertAlias	String	Yes	Both		
UserName	UserName	String	Yes	Both		
Password	Password	Password	Yes	Both		

Note: *To use HTTPS, two environment components require nondefault settings: in the integration server, the Web server configuration must have its "Enable SSL" parameter set to "True", and in the HTTP(S) external, the SSL configuration must supply a value for its "TrustStore" parameter.*

4.1.5. SMTP

For Simple Mail Transfer Protocol, the e-mail eWay uses the SMTP transport protocol to send and receive e-mail.

Table 7 lists the attributes of the SMTP TAD and corresponding OTD.

Table 7 Attributes for the SMTP Transport Attributes Definition

Name	Display	Type	Req?	Direction	Default	List
SenderAddress	SenderAddress	String	Yes	ToPartner		
Host	Host	String	Yes	ToPartner		
PortNumber	PortNumber	String	Yes	ToPartner		
UserName	UserName	String	No	Both		
Password	Password	Password	No	Both		

4.1.6. ChannelManagerFile

The ChannelManagerFile transport attributes definition is the same as the FILE TAD, with the addition of Channel Manager functionality (trading partner lookup, tracking, request/response, and so forth). For more information, see **"FILE" on page 40** and/or **"Channel Manager" on page 44**.

Table 8 lists the attributes of the ChannelManagerFile TAD and corresponding OTD.

Table 8 Attributes for the ChannelManagerFile TAD

Name	Display	Type	Req?	Directn	Default	List
ChannelManagerMode	ChannelManagerMode	List of Values	Yes	Both	FILE	FILE
FilePattern	FilePattern	String	Yes	Both		
Directory	Directory	String	Yes	Both		
PollMilliSeconds	PollMilliSeconds	Integer	Yes	Both		

4.1.7. ChannelManagerFTP

The ChannelManagerFTP transport attributes definition is the same as the FTP TAD, with the addition of Channel Manager functionality (trading partner lookup, tracking, request/response, and so forth). For more information, see **"FTP" on page 40** and/or **"Channel Manager" on page 44**.

Table 9 lists the attributes of the ChannelManagerFTP TAD and corresponding OTD.

Table 9 Attributes for the ChannelManagerFTP TAD

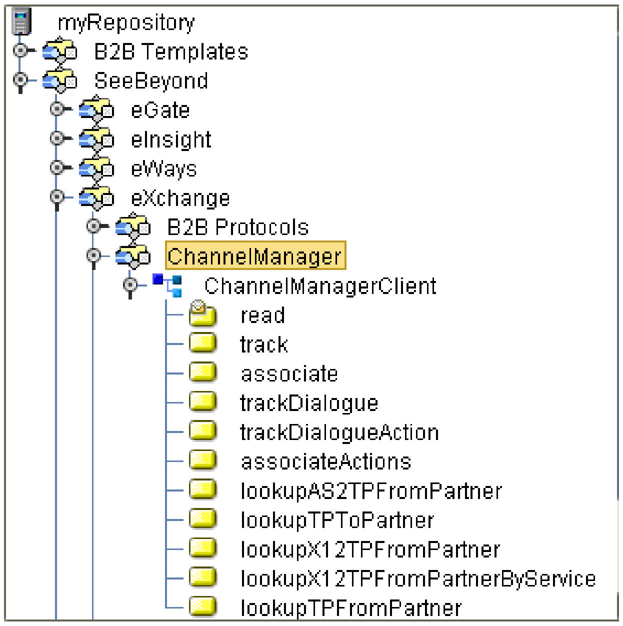
Name	Display	Type	Req?	Directn	Default	List
ChannelManagerMode	ChannelManagerMode	List of Values	Yes	Both	FTP	FTP
FilePattern	FilePattern	String	Yes	Both		
Directory	Directory	String	Yes	Both		
UserName	UserName	String	Yes	Both		
Password	Password	Password	Yes	Both		
HostName	HostName	String	Yes	Both		
PortNumber	PortNumber	Integer	No	Both		
SocksEnabled	SocksEnabled	Boolean	No	Both	false	
SocksHostName	SocksHostName	String	No	Both		
SocksUserName	SocksUserName	String	No	Both		
SocksPassword	SocksPassword	Password	No	Both		
SocksServerPort	SocksServerPort	String	No	Both		

4.2 Channel Manager

The Channel Manager facility provides several services to access or write information in the eXchange database. It tracks messages and packages, associates responses to requests and tracks them, and retrieves trading partner information.

The SeeBeyond > eXchange > **ChannelManager** folder (see Figure 15) contains the ChannelManagerClient OTD.

Figure 15 eXchange > ChannelManager Folder



In this section

- **associate** on page 45
- **associateActions** on page 45
- **lookupAS2TPFromPartner** on page 46
- **lookupTPFromPartner** on page 46
- **lookupTPToPartner** on page 47, with delivery channel profile (DCP) containers:
  - **DCP Output Containers: lookupTPToPartner... ActionBindingProfile** on page 47
  - **DCP Output Containers: lookupTPToPartner... ToPartnerTransport** on page 51
  - **DCP Output Containers: lookupTPToPartner... FromPartnerTransport** on page 52
  - **DCP Output Containers: lookupTPToPartner... ToPartnerPackager** on page 53
  - **DCP Output Containers: lookupTPToPartner... FromPartnerUnpackager** on page 56
- **read** on page 59
- **track** on page 59
- **trackDialogue** on page 60
- **trackDialogueAction** on page 61

associate

ChannelManagerClient.**associate** is used to associate a response to a request. This operation can only be used for message level documents—in other words envelopes, as opposed to business documents.

The service associates the response to the request using a message identifier to tie the two messages to each other.

Table 10 Input Containers for ChannelManagerClient.associate

Name	Description
OrigPkgHdrId	Database ID of the original message
AckPkgHeaderId	Database ID of the acknowledgement message
PkgType	Name of the messaging or packaging envelope used for the message, such as **ISA** or **GS**.
TPId	The database's unique ID for the trading partner; in other words, the foreign key to ex_trading_partner.
MsgUniqId	Unique ID for the message.
ErrorFlag	A value of **Y** signifies that the message contains a "business" type of error: could not decrypt, could not verify signature, and so forth.
ErrorNo	*(reserved)*
ErrorStr	A description of the error.

Table 11 Output Container for ChannelManagerClient.associate

Name	Description
PkgAssocId	Association ID used to associate the response package to the request package.

associateActions

ChannelManagerClient.**associateActions** is similar to the associate operation, in that it associates a document response to a document request (for example, in X12, a 997 or 855 response to an original 850 request).

Table 12 Input Containers for ChannelManagerClient.associateActions

Name	Description
OrigPkgHdrId	Database ID of the original message
AckPkgHeaderId	Database ID of the acknowledgment message
PkgType	Name of the messaging or packaging envelope used for the message, such as **ISA** or **GS**.

Table 12 Input Containers for ChannelManagerClient.associateActions (Continued)

Name	Description
TPId	The database's unique ID for the trading partner; in other words, the foreign key to ex_trading_partner.
MsgUniqId	Unique ID for the message.
ErrorFlag	A value of **Y** signifies that the message contains a "business" type of error: could not decrypt, could not verify signature, and so forth.
ErrorNo	*(reserved)*
ErrorStr	A description of the error.

Table 13 Output Container for ChannelManagerClient.associateActions

Name	Description
isAssociated	A value of **Y** signifies that an associated action exists.

lookupAS2TPFromPartner

ChannelManagerClient.**lookupAS2TPFromPartner** is used to fetch trading partner information (the delivery channel profile, or DCP) from an inbound AS2 message. For example, if an HTTP server gets a Post, it can be used to determine which trading partner sent the message and to look up parameters associated with that partner.

This is not a generic operation; its use is reserved for SeeBeyond only at this time.

lookupTPFromPartner

ChannelManagerClient.**lookupTPFromPartner** is used to fetch trading partner information (the delivery channel profile, or DCP) from a generic inbound message.

Table 14 Input Containers for ChannelManagerClient.lookupTPFromPartner

Name	Description
PartnerName	Name of the trading partner whose information is to be fetched.
ServiceName	Name of the messaging service being used to process the message.
ActionName	Name of the messaging action being used to process the message.

Table 15 Output Containers for ChannelManagerClient.lookupTPFromPartner

Name	Description
text	See output containers for **"lookupTPToPartner" on page 47**, especially the extensive descriptions in Table 17 and Tables 18 through 22.

lookupTPToPartner

ChannelManagerClient.**lookupTPToPartner** is used to fetch trading partner information (the delivery channel profile, or DCP) from a generic inbound message.

Table 16 Input Containers for ChannelManagerClient.lookupTPToPartner

Name	Description
PartnerName	Name of the trading partner whose information is to be fetched; used only if TPProfileID is null (see next line).
TPProfileId	Trading partner profile ID to be used. If TPProfileID exists, then the TP lookup operation takes place using the TPProfileID, action name, and service name. If TPProfileID is null, lookup uses the partner name.
ServiceName	Name of the messaging service being used to process the message.
ActionName	Name of the messaging action being used to process the message.

Table 17 DCP Root Output Containers for lookupTPToPartner

Name	Description
ChannelId	*(not currently used)*
Retries	The maximum number of retries permitted.
RetryInterval	The interval between retries.
TradingPartnerId	*(not currently used)*
HostGenId	eXchange-generated unique ID identifying the B2B host.
TPGenId	eXchange-generated unique ID identifying the trading partner.
ActionBindingProfile	See Table 18: **DCP Output Containers: lookupTPToPartner... ActionBindingProfile** on page 47.
ToPartnerTransport	See Table 19: **DCP Output Containers: lookupTPToPartner... ToPartnerTransport** on page 51.
FromPartnerTransport	See Table 20: **DCP Output Containers: lookupTPToPartner... FromPartnerTransport** on page 52.
ToPartnerPackager	See Table 21: **DCP Output Containers: lookupTPToPartner... ToPartnerPackager** on page 53.
FromPartnerUnpackager	See Table 22: **DCP Output Containers: lookupTPToPartner... FromPartnerUnpackager** on page 56.

Table 18 DCP Output Containers: lookupTPToPartner... ActionBindingProfile

Name	Description
CPAId	Trading profile ID that unique identifies a trading partner agreement.

Table 18 DCP Output Containers: lookupTPToPartner... ActionBindingProfile (Continued)

Name	Description
ActionId	*(not currently used)*
ActionName	Name of the business transaction associated with this delivery channel.
ToPartnerFlag	A value of **TRUE** signifies that this message is outbound to the trading partner.
ServiceName	Name of the messaging service being used to deliver the message.
RoleName	Name of the role that the trading partner assumes in the CPA.
HostId	Party ID identifying the B2B host to the trading partner.
Start	CPA effective data.
End	CPA expiration date.
ConcurrentConversations	The maximum number of concurrent conversations allowed by the CPA.
InvocationLimit	The maximum number of conversations that can be processed under the CPA.
ProtocolName	Name of the protocol being used to deliver the message.
ProtocolVersion	Version of the protocol being used to deliver the message.
IsNonRepudiationRequired	A value of **TRUE** signifies that nonrepudiation is required and a digital signature is to be used.
IsConfidential	A value of **TRUE** signifies that the message is to be encrypted.
IsAuthenticated	A value of **TRUE** signifies that the delivery channel requires authentication of the sender of the message before the message is delivered.
IsAuthorizationRequired	*(not currently used)*
IsTamperProof	*(not currently used)*
IsIntelligibleCheckRequired	*(not currently used)*
IsSecureTransportRequired	*(not currently used)*
BatchType	*(not currently used)*
Operation	*(not currently used)*
GeneratedPortType	*(not currently used)*
GeneratedPortTypeNS	*(not currently used)*
ServiceExtAttrValueXML	*(not currently used)*
ValidationExtAttrValueXML	*(not currently used)*
EnvelopeExtAttrValueXML	*(not currently used)*
InternalSenderTransport ExtAttrValueXML	TAD to be used when sending messages from eXchange to the internal system.

Table 18 DCP Output Containers: lookupTPToPartner... ActionBindingProfile (Continued)

Name	Description
InternalReceiverTransport ExtAttrValueXML	TAD to be used when receiving messages into eXchange from the internal system.
InternalReceiverTransport ProtocolName	Name of transport protocol to be used for receiving messages into eXchange when using the internal delivery channel.
InternalReceiverTransport ProtocolVersion	Version of transport protocol to be used for sending messages from eXchange when using the internal delivery channel.
InternalSenderTransport ProtocolName	Name of transport protocol to be used for sending messages from eXchange when using the internal delivery channel.
InternalSenderTransport ProtocolVersion	Version of transport protocol to be used for sending messages from eXchange when using the internal delivery channel.
PartyId >**PartyId**	Unique ID for trading partner, as set in trading partner configuration.
PartyId >**PartyType**	Type of unique ID for trading partner, as set in trading partner configuration.
PartyRef >**XlinkHrefName**	*(ebXML only)* Link to a location containing reference information about the trading partner.
PartyRef >**PartyRefXlinkType**	*(ebXML only)* At this time, always set to **SIMPLE**.
PartyRef >**PartyRefType**	*(ebXML only)* Document type identifier for external reference information about the trading partner.
PartyRef >**SchemaLocation**	*(ebXML only)* Schema location for the PartyRefXlinkType.
MimePackaging >**Id**	*(ebXML only)* Unique ID that identifies the MIME packaging details to be used.
MimePackaging >**Parsed**	*(ebXML only)* A value of **TRUE** signifies that the packaging constructs specified in the other child elements can be produces as well as processed at the software messaging service layer.
MimePackaging >**Generated**	*(ebXML only)* A value of **TRUE** signifies that the packaging constructs specified in the other child elements can be produces as well as processed at the software messaging service layer.
MimePackaging >CompositeList > **Encapsulation >Id**	Encapsulation identifier.
MimePackaging >CompositeList **Encapsulation >MimeType**	Value of the MIME content type for this message part (for example, **application/pkcs7-mime**)
MimePackaging >CompositeList **Encapsulation >MimeParameters**	Values of any significant MIME parameters needed to understand the processing demands of the content type.
MimePackaging >CompositeList Encapsulation > Constituent > **ExcludeFromSignature**	A value of **TRUE** signifies that the message is not to be signed.

Table 18 DCP Output Containers: lookupTPToPartner... ActionBindingProfile (Continued)

Name	Description
MimePackaging > CompositeList Encapsulation > Constituent > **MinOccurs**	*(not currently used)*
MimePackaging > CompositeList Encapsulation > Constituent > **MaxOccurs**	*(not currently used)*
MimePackaging > CompositeList Encapsulation > Constituent > **SignatureTransforms**	*(not currently used)*
MimePackaging > CompositeList Encapsulation > Constituent > **EncryptionTransforms**	*(not currently used)*
MimePackaging > CompositeList Encapsulation > Constituent > **Role**	*(not currently used)*
MimePackaging > CompositeList Encapsulation > Constituent > **MimeType**	Value of the MIME content type for this message constituent (for example, **application/pkcs7-mime**)
MimePackaging > CompositeList Encapsulation > Constituent > **Id**	*(ebXML)* Reference to the Simple Part ID.
MimePackaging > CompositeList Encapsulation > Constituent > **SimplePart > Id**	Simple Part unique ID.
MimePackaging > CompositeList Encapsulation > Constituent > **SimplePart > Role**	*(not currently used)*
MimePackaging > CompositeList Encapsulation > Constituent > **SimplePart > MimeType**	Value of the MIME content type for this message constituent (for example, **application/xml**)
MimePackaging > **Composite > Id**	Composite identifier.
MimePackaging > **Composite > MimeType**	Value of the MIME content type for this message part (for example, **multipart/related** or **multipart/signed**)
MimePackaging > **Composite > MimeParameters**	Values of any significant MIME parameters needed to understand the processing demands of the content type.
MimePackaging > Composite > **Constituent > ExcludeFromSignature**	A value of **TRUE** signifies that the message is not to be signed.
MimePackaging > Composite > **Constituent > MinOccurs**	*(not currently used)*
MimePackaging > Composite > **Constituent > MaxOccurs**	*(not currently used)*

Table 18 DCP Output Containers: lookupTPToPartner... ActionBindingProfile (Continued)

Name	Description
MimePackaging > Composite > **Constituent > SignatureTransforms**	*(not currently used)*
MimePackaging > Composite > **Constituent > EncryptionTransforms**	*(not currently used)*
MimePackaging > Composite > **Constituent > Role**	*(not currently used)*
MimePackaging > Composite > **Constituent > MimeType**	Value of the MIME content type for this message constituent (for example, **application/xml**)
MimePackaging > Composite > **Constituent > Id**	*(ebXML)* Reference to the Simple Part ID.
MimePackaging > Composite > **Constituent > SimplePart > Id**	Simple Part unique ID.
MimePackaging > Composite > **Constituent > SimplePart > Role**	*(not currently used)*
MimePackaging > Composite > **Constituent > SimplePart > MimeType**	Value of the MIME content type for this message constituent (for example, **application/xml**)

Table 19 DCP Output Containers: lookupTPToPartner... ToPartnerTransport

Name	Description
TransportProtocolName	Name of transport protocol to be used for sending to a partner from eXchange.
TransportProtocolVersion	Version of transport protocol to be used for sending to a partner from eXchange.
ServerCertificateRef	*(not currently used)*
ClientSecurityDetailsRef	*(not currently used)*
SecurityProtocolName	Name of transport security protocol to be used for transport to the trading partner.
SecurityProtocolVersion	Version of transport security protocol to be used transport to the trading partner.
ClientCertificateRef	*(not currently used)*
ServerSecurityDetailsRef	*(not currently used)*
TransportExtAttrValueXML	Transport Attributes Definition (TAD) XML to be used when sending to a trading partner from eXchange.
SecurityExtAttrValueXML	*(not currently used)*
EncryptionAlgorithm > **Name**	Name of the encryption algorithm to be used when sending to a trading partner from eXchange.
EncryptionAlgorithm > **OID**	Object identifier for the encryption algorithm.
EncryptionAlgorithm > **EnumerationType**	*(not currently used)*

Table 19 DCP Output Containers: lookupTPToPartner... ToPartnerTransport (Continued)

Name	Description
EncryptionAlgorithm > **MinimumStrength**	*(not currently used)*
EncryptionAlgorithm > **W3C**	*(not currently used)*
EndPoint > **Type**	Type of endpoint specified for the trading partner.
EndPoint > **URI**	Uniform Resource Identifier specified for the trading partner.
AccessAuthentication	*(not currently used)*
TransportExtAttributeValue > **Attribute > name**	*(not currently used)*
TransportExtAttributeValue > **Attribute > value**	*(not currently used)*
SecurityExtAttributeValue > **Attribute > name**	*(not currently used)*
SecurityExtAttributeValue > **Attribute > value**	*(not currently used)*
TransportExtAttrValueXML	TAD attribute values.
SecurityExtAttrValueXML	*(not currently used)*

Table 20 DCP Output Containers: lookupTPToPartner... FromPartnerTransport

Name	Description
TransportProtocolName	Name of transport protocol to be used for receiving into eXchange from a trading partner.
TransportProtocolVersion	Version of transport protocol to be used for receiving into eXchange from a trading partner.
ServerCertificateRef	*(not currently used)*
ClientSecurityDetailsRef	*(not currently used)*
SecurityProtocolName	Name of transport security protocol to be used for transport to the trading partner.
SecurityProtocolVersion	Version of transport security protocol to be used transport to the trading partner.
ClientCertificateRef	*(not currently used)*
ServerSecurityDetailsRef	*(not currently used)*
TransportExtAttrValueXML	Transport Attributes Definition (TAD) XML to be used when receiving into eXchange from a trading partner.
SecurityExtAttrValueXML	*(not currently used)*
EncryptionAlgorithm > **Name**	Name of the encryption algorithm to be used when receiving into eXchange from a trading partner.
EncryptionAlgorithm > **OID**	Object identifier for the encryption algorithm.
EncryptionAlgorithm > **EnumerationType**	*(not currently used)*

Table 20 DCP Output Containers: lookupTPToPartner... FromPartnerTransport (Continued)

Name	Description
EncryptionAlgorithm > **MinimumStrength**	*(not currently used)*
EncryptionAlgorithm >**W3C**	*(not currently used)*
Endpoint >Type	Type of endpoint specified for the trading partner.
Endpoint >URI	Uniform Resource Identifier specified for the trading partner.
AccessAuthentication	*(not currently used)*
TransportExtAttributeValue > **Attribute >name**	*(not currently used)*
TransportExtAttributeValue > **Attribute >value**	*(not currently used)*
SecurityExtAttributeValue > **Attribute >name**	*(not currently used)*
SecurityExtAttributeValue > **Attribute >value**	*(not currently used)*
TransportExtAttrValueXML	TAD attribute values.
SecurityExtAttrValueXML	*(not currently used)*

Table 21 DCP Output Containers: lookupTPToPartner... ToPartnerPackager

Name	Description
Level	*(not currently used)*
EncryptionCertificateRef	Reference to the trading partner's encryption certificate to be used, located in the keystore of the B2B host.
HashFunction	Name of the hash algorithm to be used.
SigningSecurityDetailsRef	Name of the security truststore used for signature.
PackPipeline	Packaging protocol (process) to be used.
PersistDuration	Duration specified for the transaction to be stored and available for review.
EncryptionSecurityDetailsRef	Name of the security truststore used for encryption.
SigningCertificateRef	Reference to the B2B host signature key to be used, located in the keystore of the B2B host.
SyncReplyMode	Message response type (either **SYNC** or **ASYNC**).
MessageOrderSemantics	*(not currently used)*
AckRequested	Specifies to the trading partner that an acknowledgment is requested.
AckSignatureRequested	Specifies to the trading partner that a signature is requested with the acknowledgment.
DuplicateElimination	If set, indicates that the trading partner is to check for a duplicate message.

Table 21 DCP Output Containers: lookupTPToPartner... ToPartnerPackager (Continued)

Name	Description
Actor	*(not currently used)*
PackagingProtocolName	Name of the enveloping protocol to be used to send to a trading partner.
PackagingProtocolVersion	Version of the enveloping protocol to be used to send to a trading partner.
DigitalEnvelopeProtocolName	Name of the signature algorithm to be used to send to a trading partner.
DigitalEnvelopeProtocolVersion	Version of the signature algorithm to be used to send to a trading partner.
NonRepudiationProtocolName	Name of the encryption algorithm to be used to encrypt the message.
NonRepudiationProtocolVersion	Version of the encryption algorithm to be used to encrypt the message.
EnvelopeExtAttrValueXML	Enveloping Attributes Definition (EAD to be used when sending to a trading partner.
EnvelopeExtAttributeValue > Attribute >**name**	*(not currently used)*
EnvelopeExtAttributeValue > Attribute >**value**	*(not currently used)*
EnvelopeExtAttributeDef > Attribute >**name**	*(not currently used)*
EnvelopeExtAttributeDef > Attribute >**displayname**	*(not currently used)*
EnvelopeExtAttributeDef > Attribute >**type**	*(not currently used)*
EnvelopeExtAttributeDef > Attribute >**maxChars**	*(not currently used)*
EnvelopeExtAttributeDef > Attribute >**minChars**	*(not currently used)*
EnvelopeExtAttributeDef > Attribute >**requiredFlag**	*(not currently used)*
EnvelopeExtAttributeDef > Attribute >**fixedFlag**	*(not currently used)*
EnvelopeExtAttributeDef > Attribute >**default**	*(not currently used)*
EnvelopeExtAttributeDef > Attribute >**direction**	*(not currently used)*
EnvelopeExtAttributeDef > Attribute >**mergeOption**	*(not currently used)*
EnvelopeExtAttributeDef > Attribute >**FormatString**	*(not currently used)*

Table 21 DCP Output Containers: lookupTPToPartner... ToPartnerPackager (Continued)

Name	Description
EnvelopeExtAttributeDef > Attribute > **List**	*(not currently used)*
ToPartnerPackProtocolSubVersion > **SubVersion**	*(not currently used)*
ToPartnerPackProtocolSubVersion > **ExtAttrValueXML**	*(not currently used)*
ToPartnerPackProtocolSubVersion > ExtAttributeDef > Attribute > **name**	*(not currently used)*
ToPartnerPackProtocolSubVersion > ExtAttributeDef > Attribute > **displayname**	*(not currently used)*
ToPartnerPackProtocolSubVersion > ExtAttributeDef > Attribute > **type**	*(not currently used)*
ToPartnerPackProtocolSubVersion > ExtAttributeDef > Attribute > **maxChars**	*(not currently used)*
ToPartnerPackProtocolSubVersion > ExtAttributeDef > Attribute > **minChars**	*(not currently used)*
ToPartnerPackProtocolSubVersion > ExtAttributeDef > Attribute > **requiredFlag**	*(not currently used)*
ToPartnerPackProtocolSubVersion > ExtAttributeDef > Attribute > **fixedFlag**	*(not currently used)*
ToPartnerPackProtocolSubVersion > ExtAttributeDef > Attribute > **default**	*(not currently used)*
ToPartnerPackProtocolSubVersion > ExtAttributeDef > Attribute > **direction**	*(not currently used)*
ToPartnerPackProtocolSubVersion > ExtAttributeDef > Attribute > **mergeOption**	*(not currently used)*
ToPartnerPackProtocolSubVersion > ExtAttributeDef > Attribute > **FormatString**	*(not currently used)*
ToPartnerPackProtocolSubVersion > ExtAttributeDef > Attribute > **List**	*(not currently used)*
ToPartnerPackProtocolSubVersion > ExtAttributeValue > Attribute > **name**	*(not currently used)*

Table 21 DCP Output Containers: lookupTPToPartner... ToPartnerPackager (Continued)

Name	Description
ToPartnerPackProtocolSubVersion > ExtAttributeValue >Attribute > **value**	*(not currently used)*
ToPartnerSignal > **SignalName**	*(not currently used)*
ToPartnerSignal > **ExtAttributeValue**	*(not currently used)*
EncryptionAlgorithm > **Name**	*(not currently used)*
EncryptionAlgorithm > **OID**	*(not currently used)*
EncryptionAlgorithm > **EnumerationType**	*(not currently used)*
EncryptionAlgorithm > **MinimumStrength**	*(not currently used)*
EncryptionAlgorithm > **W3C**	*(not currently used)*
SignatureAlgorithm > **Name**	*(not currently used)*
SignatureAlgorithm > **OID**	*(not currently used)*
SignatureAlgorithm > **EnumerationType**	*(not currently used)*
SignatureAlgorithm > **W3C**	*(not currently used)*

Table 22 DCP Output Containers: lookupTPToPartner... FromPartnerUnpackager

Name	Description
Level	*(not currently used)*
EncryptionCertificateRef	Reference to the trading partner's decryption private key to be used, located in the keystore of the B2B host.
HashFunction	Name of the hash algorithm to be used.
SigningSecurityDetailsRef	Name of the security truststore used for verification.
UnpackPipeline	Packaging protocol (process) to be used.
PersistDuration	Duration specified for the transaction to be stored and available for review.
EncryptionSecurityDetailsRef	Name of the security truststore used for encryption.
SigningCertificateRef	Reference to the trading partner's signature certificate to be used, located in the keystore of the B2B host.
SyncReplyMode	Message response type (either **SYNC** or **ASYNC**).
MessageOrderSemantics	*(not currently used)*
AckRequested	Specifies that the trading partner has requested an acknowledgment.
AckSignatureRequested	Specifies that the trading partner has requested a signature with the acknowledgment.

Table 22 DCP Output Containers: lookupTPToPartner... FromPartnerUnpackager (Continued)

Name	Description
DuplicateElimination	If set, indicates that eXchange is to check for a duplicate message.
Actor	*(not currently used)*
PackagingProtocolName	Name of the enveloping protocol to be used to send to a trading partner.
PackagingProtocolVersion	Version of the enveloping protocol to be used to send to a trading partner.
DigitalEnvelopeProtocolName	Name of the signature algorithm to be used to verify the message.
DigitalEnvelopeProtocolVersion	Version of the signature algorithm to be used to verify the message.
NonRepudiationProtocolName	Name of the encryption algorithm to be used to decrypt the message.
NonRepudiationProtocolVersion	Version of the encryption algorithm to be used to decrypt the message.
ValidationExtAttrValueXML	Enveloping Attributes Definition (EAD to be used when receiving from a trading partner.
ValidationExtAttributeValue > Attribute >**name**	*(not currently used)*
ValidationExtAttributeValue > Attribute >**value**	*(not currently used)*
ValidationExtAttributeDef > Attribute >**name**	*(not currently used)*
ValidationExtAttributeDef > Attribute >**displayname**	*(not currently used)*
ValidationExtAttributeDef > Attribute >**type**	*(not currently used)*
ValidationExtAttributeDef > Attribute >**maxChars**	*(not currently used)*
ValidationExtAttributeDef > Attribute >**minChars**	*(not currently used)*
ValidationExtAttributeDef > Attribute >**requiredFlag**	*(not currently used)*
ValidationExtAttributeDef > Attribute >**fixedFlag**	*(not currently used)*
ValidationExtAttributeDef > Attribute >**default**	*(not currently used)*
ValidationExtAttributeDef > Attribute >**direction**	*(not currently used)*
ValidationExtAttributeDef > Attribute >**mergeOption**	*(not currently used)*

Table 22 DCP Output Containers: lookupTPToPartner... FromPartnerUnpackager (Continued)

Name	Description
ValidationExtAttributeDef > Attribute > **FormatString**	*(not currently used)*
ValidationExtAttributeDef > Attribute > **List**	*(not currently used)*
FromPartnerPackProtocolSubVersion > **SubVersion**	*(not currently used)*
FromPartnerPackProtocolSubVersion > **ExtAttrValueXML**	*(not currently used)*
FromPartnerPackProtocolSubVersion > ExtAttributeDef > Attribute > **name**	*(not currently used)*
FromPartnerPackProtocolSubVersion > ExtAttributeDef > Attribute > **displayname**	*(not currently used)*
FromPartnerPackProtocolSubVersion > ExtAttributeDef > Attribute > **type**	*(not currently used)*
FromPartnerPackProtocolSubVersion > ExtAttributeDef > Attribute > **maxChars**	*(not currently used)*
FromPartnerPackProtocolSubVersion > ExtAttributeDef > Attribute > **minChars**	*(not currently used)*
FromPartnerPackProtocolSubVersion > ExtAttributeDef > Attribute > **requiredFlag**	*(not currently used)*
FromPartnerPackProtocolSubVersion > ExtAttributeDef > Attribute > **fixedFlag**	*(not currently used)*
FromPartnerPackProtocolSubVersion > ExtAttributeDef > Attribute > **default**	*(not currently used)*
FromPartnerPackProtocolSubVersion > ExtAttributeDef > Attribute > **direction**	*(not currently used)*
FromPartnerPackProtocolSubVersion > ExtAttributeDef > Attribute > **mergeOption**	*(not currently used)*
FromPartnerPackProtocolSubVersion > ExtAttributeDef > Attribute > **FormatString**	*(not currently used)*
FromPartnerPackProtocolSubVersion > ExtAttributeDef > Attribute > **List**	*(not currently used)*
FromPartnerPackProtocolSubVersion > ExtAttributeValue > Attribute > **name**	*(not currently used)*

Table 22 DCP Output Containers: lookupTPToPartner... FromPartnerUnpackager (Continued)

Name	Description
FromPartnerPackProtocolSubVersion > ExtAttributeValue >Attribute >**value**	*(not currently used)*
FromPartnerSignal > **SignalName**	*(not currently used)*
FromPartnerSignal > **ExtAttributeValue**	*(not currently used)*
EncryptionAlgorithm > **Name**	*(not currently used)*
EncryptionAlgorithm > **OID**	*(not currently used)*
EncryptionAlgorithm > **EnumerationType**	*(not currently used)*
EncryptionAlgorithm > **MinimumStrength**	*(not currently used)*
EncryptionAlgorithm > **W3C**	*(not currently used)*
SignatureAlgorithm > **Name**	*(not currently used)*
SignatureAlgorithm > **OID**	*(not currently used)*
SignatureAlgorithm > **EnumerationType**	*(not currently used)*
SignatureAlgorithm > **W3C**	*(not currently used)*

read

ChannelManagerClient.**read** performs a read operation on the server, receiving messages to the B2B host from internal delivery channels or from external trading partners.

Table 23 Output Containers for ChannelManagerClient.read

Name	Description
text	See output containers for **"lookupTPToPartner" on page 47**, especially the extensive descriptions in Table 17 and Tables 18 through 22.

track

ChannelManagerClient.**track** performs a track operation to store the message to the eXchange database

Table 24 Input Containers for ChannelManagerClient.track

Name	Description
Protocol	Name of the protocol being used to handle the message.

Table 24 Input Containers for ChannelManagerClient.track (Continued)

Name	Description
ReceiveFlag	A value of **Y** signifies that the request message was inbound.
BufferId	*ebXML only.* Conversation ID.
OrderNumInBuffer	*ebXML only.* Reserved for use in message ordering.
MsgUniqId	Unique ID for the message.
TPId	The database's unique ID for the trading partner; in other words, the foreign key to ex_trading_partner.
OrdMsgId	*(not currently used)*
Multiple Content	*(not currently used)*
PkgType	Name of the messaging or packaging envelope used for the message, such as ISA or GS.
ErrorFlag	A value of **Y** signifies that the message contains a "business" type of error: could not decrypt, could not verify signature, and so forth.
RespRequired	A value of **Y** signifies that a response to this message is required.
MsgBlob	Container for the message payload.
SignedFlag	A value of **Y** signifies that the message is signed.
CompressedFlag	A value of **Y** signifies that the message is compressed.
EncryptedFlag	A value of **Y** signifies that the message is encrypted.
MessageType	Message type for the message, such as **Message** or **Ack**.
Resendable	A value of **Y** signifies that the message can be re-sent.
Service	Service name for the request message for which the response is received.
Action	Action name for the request message for which the response is received.

Table 25 Output Container for ChannelManagerClient.track

Name	Description
MsgHdrId	Message header ID, used for message association.

trackDialogue

ChannelManagerClient.**trackDialogue** is used to write the initial message—that is, the first business document in a conversation—to message tracking. To write subsequent messages in the same conversation, the trackDialogueAction operation is used.

Table 26 Input Containers for ChannelManagerClient.trackDialogue

Name	Description
tpNetworkId	eXchange-generated unique ID identifying the trading partner.
dialogueID	Database-assigned unique ID identifying the business conversation.

Table 26 Input Containers for ChannelManagerClient.trackDialogue (Continued)

Name	Description
dialogueIdentifier	Dialog ID in the message.
serviceName	Name of the messaging service being used to handle the message.
activeFlag	A value of **Y** signifies that the business conversation is active.
Status	Status of the business conversation.
startDate	Timestamp recording when the business conversation initiated.
endDate	Timestamp recording when the business conversation terminated.
protocol	Name of the protocol being used to handle the message.
hostNetworkId	eXchange-generated unique ID identifying the B2B host.
isResponse	A value of **true** signifies that the message is a response to a previous message.

Table 27 Output Containers for ChannelManagerClient.trackDialogue

Name	Description
tpNetworkId	eXchange-generated unique ID identifying the trading partner.
dialogueID	Database-assigned unique ID identifying the business conversation.
dialogueIdentifier	Dialog ID in the message.
serviceName	Name of the messaging service being used to handle the message.

trackDialogueAction

ChannelManagerClient.**trackDialogueAction** also writes to message tracking, but it writes subsequent messagesin a business conversation (after the initial message was written by trackDialogue operation).

Table 28 Input Containers for ChannelManagerClient.trackDialogueAction

Name	Description
messageId	*(deprecated)* Duplicate of actionMessageId
actionName	Name of the messaging action that is processing the message.
receiveFlag	A value of **Y** signifies that it is an inbound message.
resendFlag	A value of **Y** signifies that this is a re-send of the message
sendCount	A value of **Y** signifies that the business conversation is active.
sequenceNum	Status of the business conversation.
referToType	*(not used)*
actStatus	*(not used)*
pkgMsgHdrId	Database-assigned unique ID for the message packaging.

Table 28 Input Containers for ChannelManagerClient.trackDialogueAction (Continued)

Name	Description
msgType	Message type for the message, such as **Message** or **Ack**.
msgEncoding	Encoding to which the message conforms.
compressedFlag	A value of **Y** signifies that the message is compressed.
encryptedFlag	A value of **Y** signifies that the message is encrypted.
envelopedFlag	A value of **Y** signifies that the message is enveloped.
signedFlag	A value of **Y** signifies that the message is signed.
msgContent	The payload of the message.
attributeMap	Extended attributes for the message.
isStoreOriginal	*(not used)* A value of **Y** signifies that the original (raw) message is to be stored in the database.
errorFlag	A value of **Y** signifies that the message has an error associated with it.
respRequiredFlag	A value of **Y** signifies that a response is required for the message.
actionMessageId	Message ID.
messageType	A value of **msg** signifies a message; **ack** signifies an acknowledgment.

Table 29 Output Containers for ChannelManagerClient.trackDialogueAction

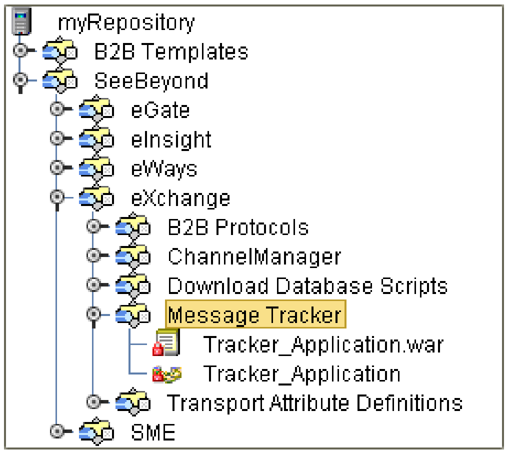
Name	Description
ActionId	ID of the service action (business transaction)
dialogueID	ID of the business dialog.

4.3 Message Tracker

The tracker application is used to record processing, packaging, and error information about messages and acknowledgments as they flow through the eXchange system.

The SeeBeyond > eXchange > **Message Tracker** folder (see Figure 16) contains two items: the tracker application itself, and its **.war** (Web archive) file.

Figure 16 eXchange > Message Tracker Folder



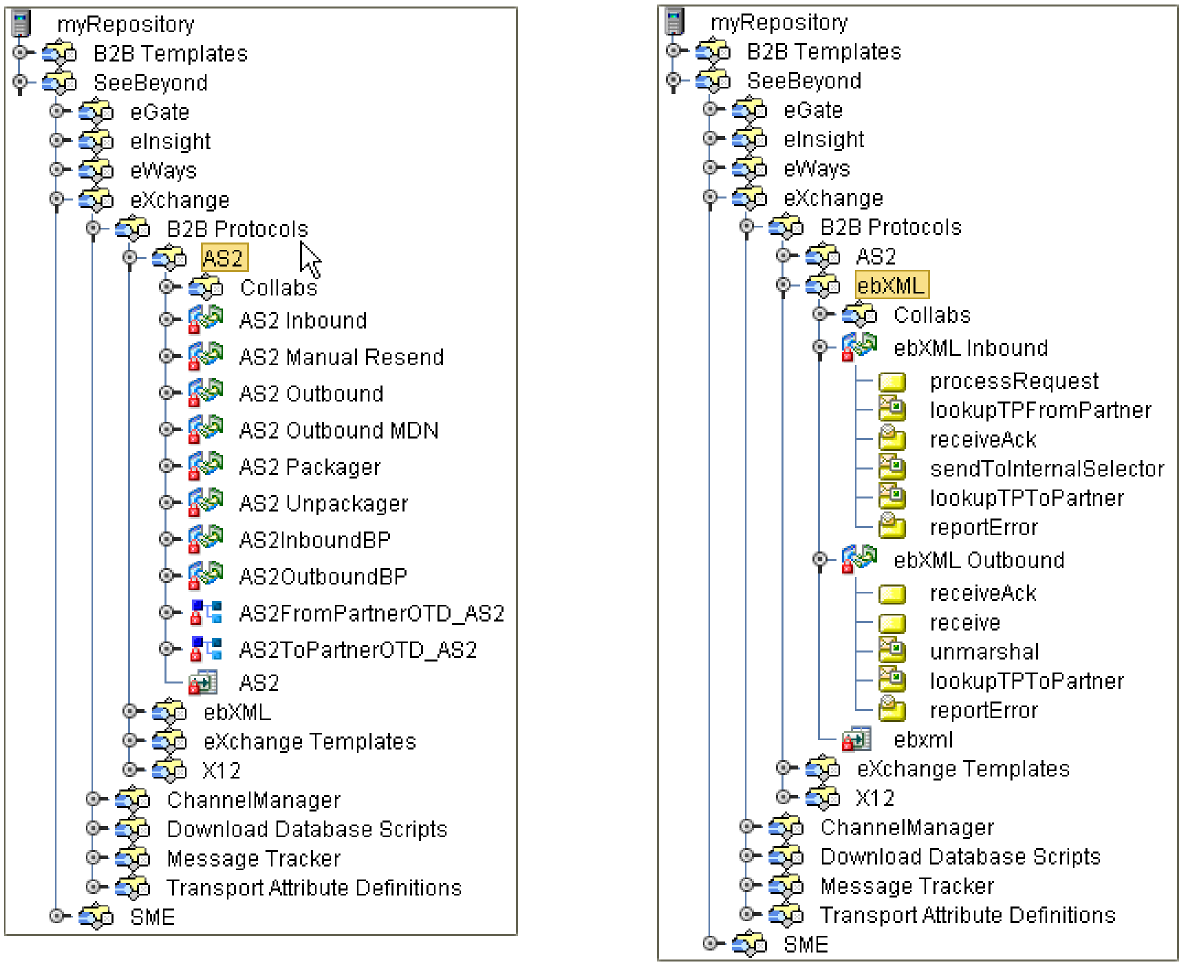
The application is entirely self-contained. All you need to do is connect an instance of the application to the B2B host you want to track and to a well-configured eXchange database. Activating this project generates an eXchange service that can be used in any other project contained in the same repository.

For information on creating and activating a connectivity map containing a tracker application, see **"Creating a Map with a B2B Host and a Message Tracker" on page 92**. For information on using the Message Tracker web client to view tracking information that has been written to the database, see **"Message Tracking" on page 122**.

4.4 B2B Protocols for AS2 and/or ebXML

If you have licensed and installed the protocol manager composite applications for other protocols, such as AS2 or ebXML, your SeeBeyond > **eXchange > B2B Protocols** folder will contain additional folders. Each additional protocol folder includes a library of collaborations, OTDs, and B2B protocols that are custom-tailored for working with AS2 and ebXML messaging. See Figure 15.

Figure 17 eXchange > B2B Protocols > AS2 and ebXML Folders



Within the SeeBeyond > eXchange > **B2B Protocols > AS2** folder are: Collaborations that provide functionality such as compression/decompression, packaging/unpackaging, Base64 encryption/decryption, encoding/decoding, signing and verifying signatures, generating/unpackaging MDNs, and more; OTDs that provide request/response functionality for each collaboration, as well as marshal/unmarshal for AS2ToPartner and AS2FromPartner messaging; and eight B2B protocol processes that encapsulate significant processing.

Within the SeeBeyond > eXchange > **B2B Protocols > ebXML** folder are: Collaborations that provide functionality for identifying, tracking, acknowledging, and validating, as well as encrypting/decrypting, signing/verifying, packing and so forth; OTDs that provide request/response functionality for each of the many collaborations; and two B2B protocol processes (one for inbound messages, one for outbound).

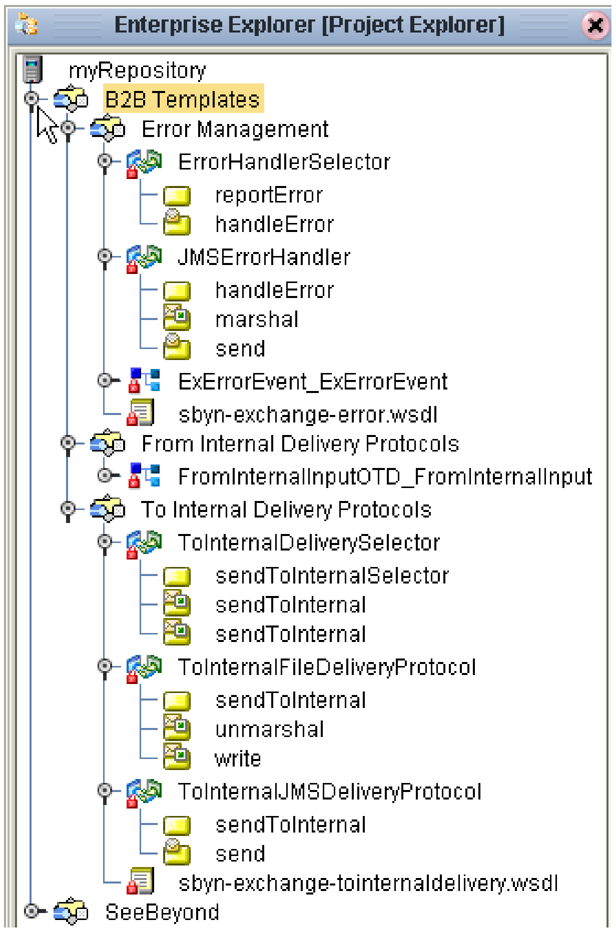
To gain a sense of how the protocol processes and collaborations/OTDs can be used, it is strongly recommended that you download and run the sample implementations. See the user's guide corresponding to the protocol manager composite application you have installed.

4.5 B2B Templates

The SeeBeyond > **B2B Templates** folder (see Figure 15) provides a set of prepackaged B2B protocols, along with OTDs and **.wsdl** files, that form the start of a library that you can create and configure to your own needs to create a customizable infrastructure.

When you export an eXchange project, the then-current version of the **B2B Templates** folder is exported along with it, allowing you keep track of the project's dependencies. Therefore, before exporting an eXchange project, you should rename your customized version of **B2B Templates** folder to something meaningful that associates it with the project(s) being exported. In that way, when you or someone else imports the project(s), there is no name clash between it and the **B2B Templates** folder in the receiving repository.

Figure 18 B2B Templates Folder



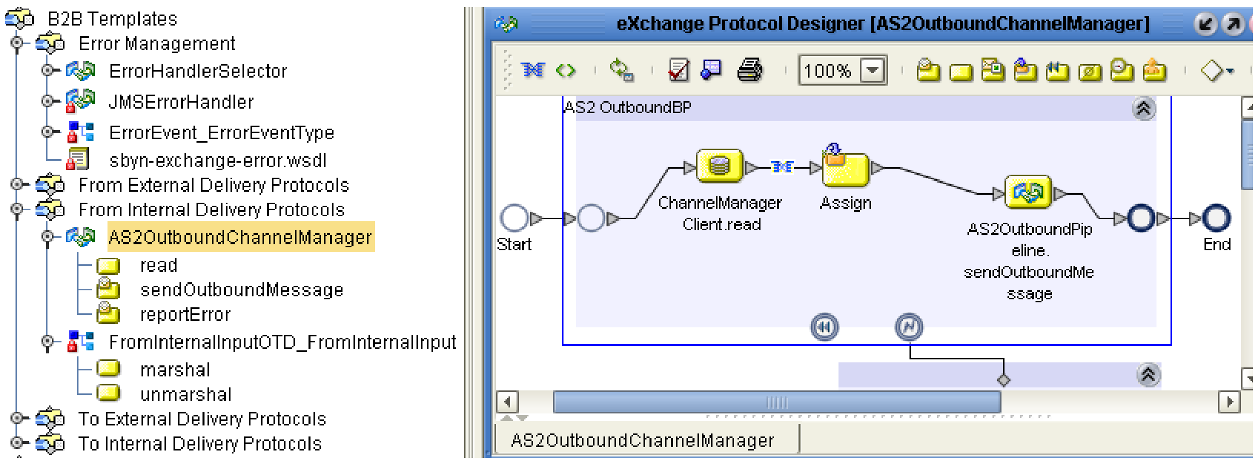
```
Enterprise Explorer [Project Explorer]
myRepository
  B2B Templates
    Error Management
      ErrorHandlerSelector
        reportError
        handleError
      JMSErrorHandler
        handleError
        marshal
        send
      ExErrorEvent_ExErrorEvent
      sbyn-exchange-error.wsdl
    From Internal Delivery Protocols
      FromInternalInputOTD_FromInternalInput
    To Internal Delivery Protocols
      ToInternalDeliverySelector
        sendToInternalSelector
        sendToInternal
        sendToInternal
      ToInternalFileDeliveryProtocol
        sendToInternal
        unmarshal
        write
      ToInternalJMSDeliveryProtocol
        sendToInternal
        send
      sbyn-exchange-tointernaldelivery.wsdl
  SeeBeyond
```

Items in the **B2B Templates** folder are interdependent not only with the pre-packaged components in the SeeBeyond > exchange folder, but also between themselves. This is illustrated in the following example.

4.5.1. Example of Interdependencies Within B2B Templates

As shipped, the **From Internal Delivery Protocols** folder contains only one item: an OTD for unmarshaling and marshaling data into and out of the B2B protocol process. When AS2 is installed as an additional protocol, however, the folder also contains a B2B protocols process itself, **AS2 OutboundChannelManager**. See Figure 19.

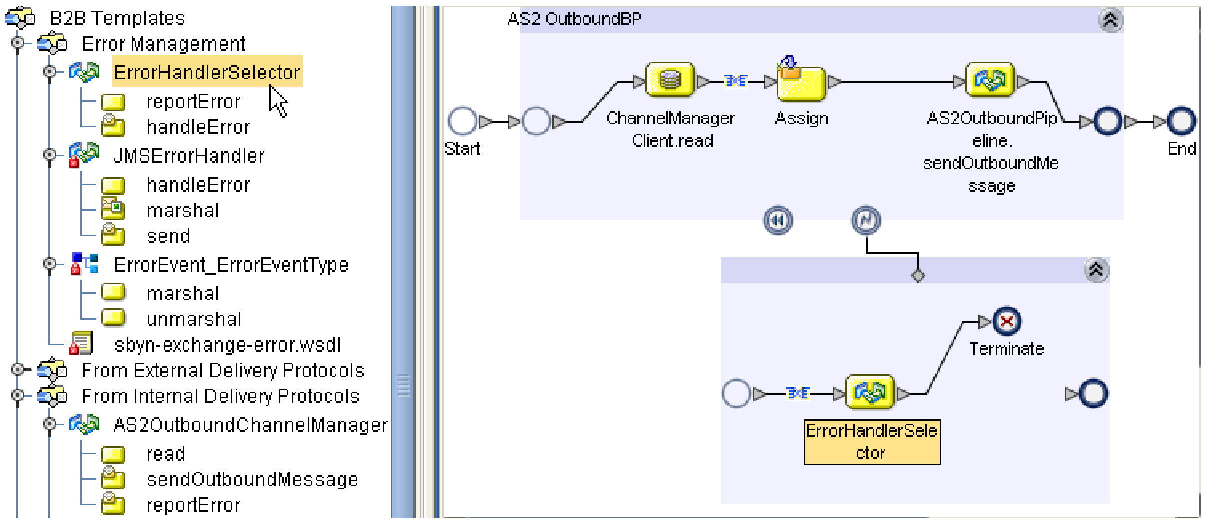
Figure 19 Template B2B Protocol Process "AS2 OutboundChannelManager"


B2B Templates
Error Management
ErrorHandlerSelector
JMSErrorHandler
ErrorEvent_ErrorEventType
sbyn-exchange-error.wsdl
From External Delivery Protocols
From Internal Delivery Protocols
AS2OutboundChannelManager
read
sendOutboundMessage
reportError
FromInternalInputOTD_FromInternalInput
marshal
unmarshal
To External Delivery Protocols
To Internal Delivery Protocols
eXchange Protocol Designer [AS2OutboundChannelManager]
100%
AS2 OutboundBP
Start
ChannelManager Client.read
Assign
AS2OutboundPipeline. sendOutboundMessage
End
AS2OutboundChannelManager

This an unmodified instance of the prepackaged AS2OutBoundBP in the SeeBeyond > eXchange > **B2B Protocols AS2** folder.

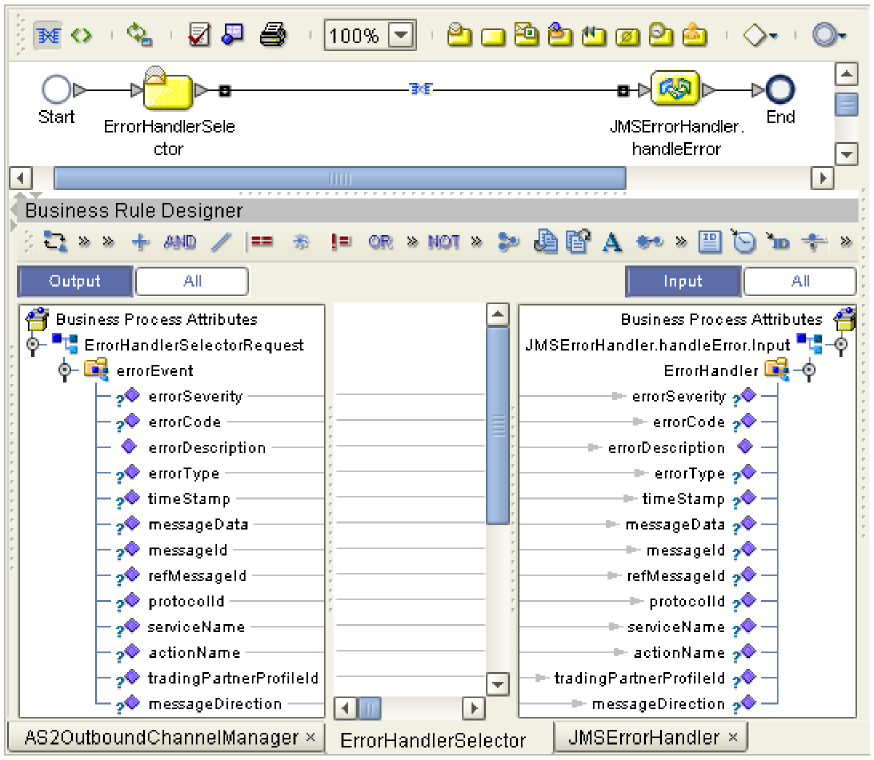
At the lower right margin of the main scope is a Catch Named Exception activity, connecting it to a scope containing an instance of a B2B protocol process from the **B2B Templates** folder: **ErrorHandlerSelector**. See Figure 20.

Figure 20 Template B2B Protocol Process "ErrorHandlerSelector"



If you open the ErrorHandlerSelector B2B protocol process, you find that by default, it uses JMS for handling errors, mapping fields under the ErrorHandlerSelectorRequest's **errorEvent** container to the handleError.Input's **ErrorHandler** container. See Figure 21.

Figure 21 Mapping from ErrorHandlerSelector to ErrorHandler



Creating a Nondefault Error Handler

The purpose of the JMSErrorHandler B2B protocol process is to send error messages to JMS, using the ErrorEvent_ErrorEventType OTD. However, in place of JMS, you could substitute SMTP to e-mail the text of the error message or FTP to write it to a file on a remote server. To create an SMTP-based error handler, for example, you would follow these steps:

1. Export the **sbyn-exchange-error.wsdl** file.
2. In the Error Management folder, create a new B2B protocol process and name it **SMTPErrorHandler** (for example).
3. Open the properties of SMTPErrorHandler and use the **WSDL** tab to load the .wsdl file you exported, the **Partners** tab to add a new partner named ErrorSelector, and the **Business Process Attributes** tab to create a new attribute named ErrorEvent (namespace **urn:sbyn-exchange-err**).
4. Drag activities onto the Protocol Designer canvas for **SMTPErrorHandler**, connect them, and configure business rules in the same way as for JMSErrorHandler.

After you have created a new error handler, you can go back to ErrorHandlerSelector and replace JMSErrorHandler with the new error handler, for example, if you wanted AS2OutboundChannelManager to deliver error messages via SMTP instead of JMS.

In the same way, you can add or modify other components in the B2B Templates folder, making them part of the toolset used by eXchange projects in this repository.

4.6 Overview of SME Processes

Although the Secure Messaging Extension (SME) is not bundled with eXchange, many of the B2B protocol processes presume that it has been installed along with eXchange. This section provides an overview of SME processes; for complete information on using SME, see the *Secure Messaging Extension User's Guide*.

In this section

4.6.1 SME Encryption/Decryption Process

This section describes the internal and external flow of the SME encryption, using the key pair encryption method.

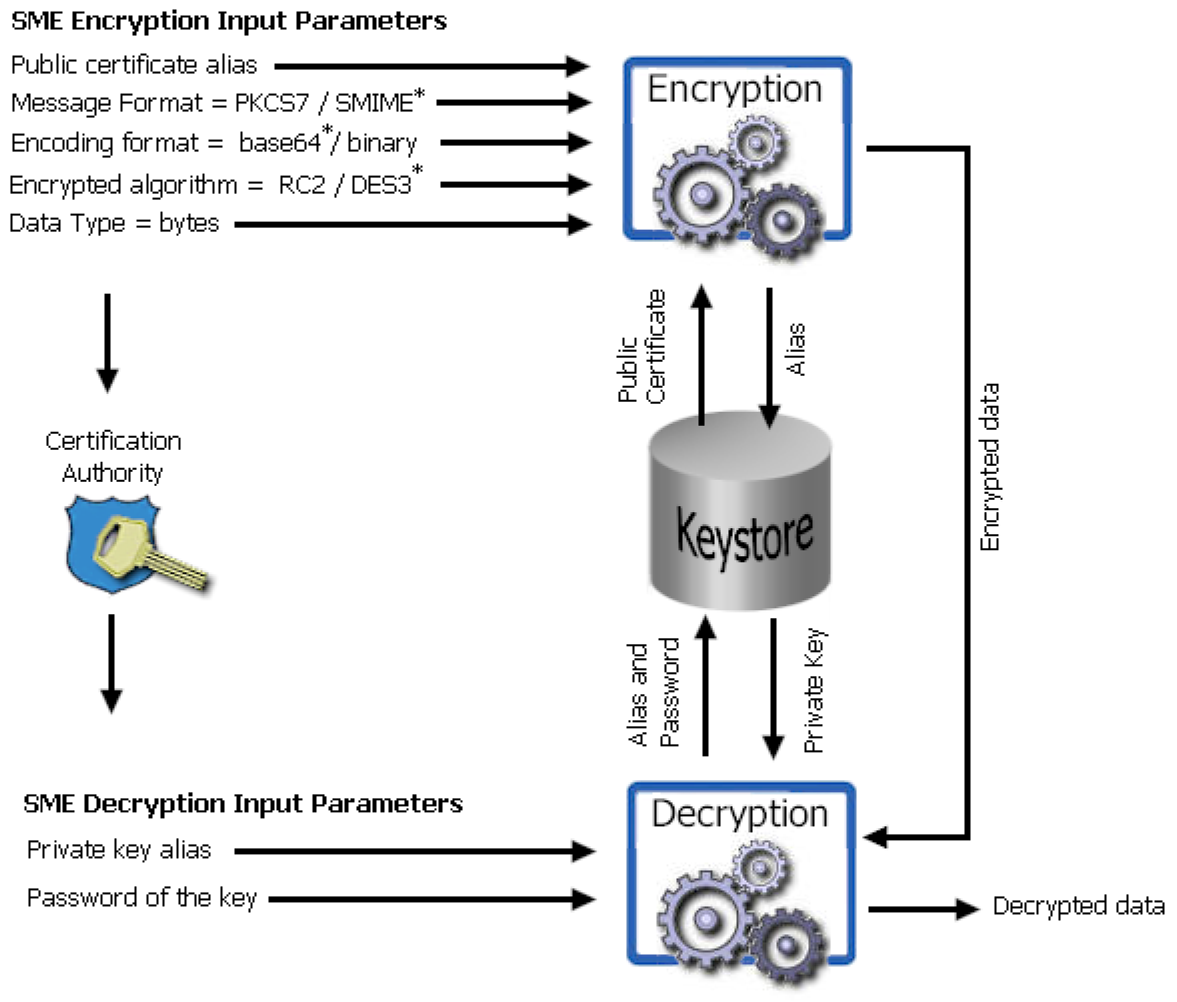
The encryption process begins when the sender's message is encrypted with the public key. The message is also signed by the sender, and the signature itself is encrypted with the sender's private key. When the reader receives the message, the encryption is decoded with the reader's private key. The sender's public certificate, located in the keystore, is used to verify the authenticity of the public key.

In addition to verifying the public key, public certificates also contain the sender's personal information, such as name, institution, and e-mail address, and are signed by a trusted Certification Authority (CA).

During encryption, a public certificate alias is used to identity the public certificate located in the keystore. During decryption, the reader's private key alias and password is used to access the private key from the keystore and decrypt the message.

The encryption/decryption process, illustrated in Figure 22, details the SME input requirements for encryption and decryption of data.

Figure 22 Encryption/Decryption Process for Secure Messaging Extension (SME)



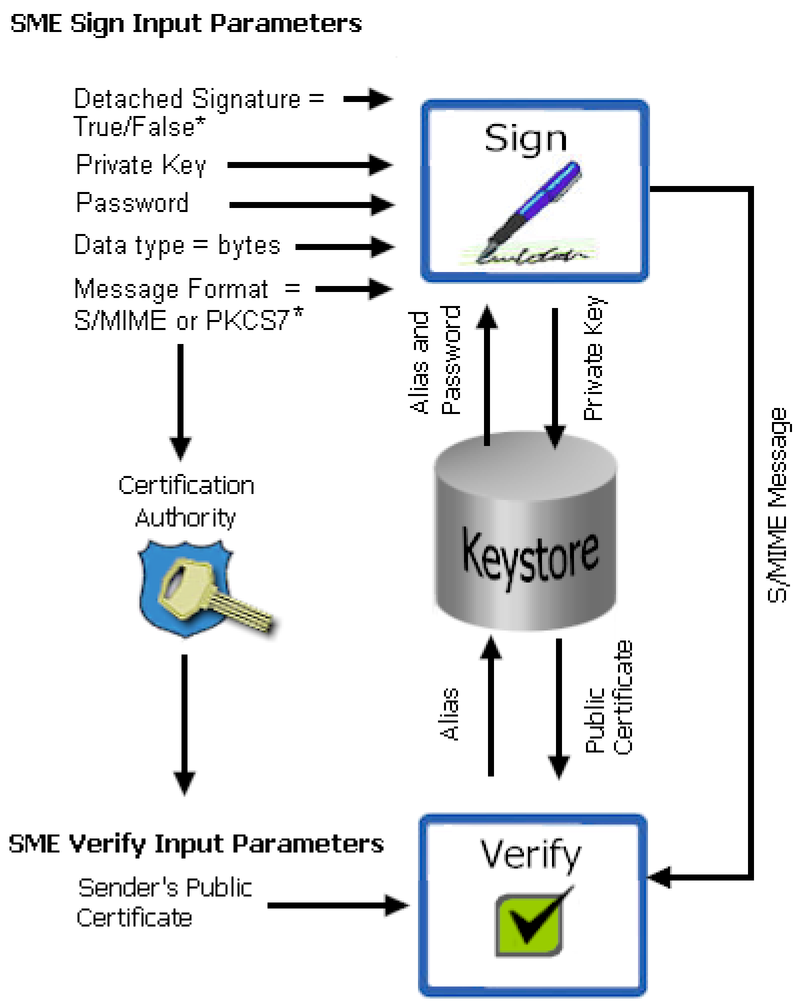
Note: *Input parameters listed with a "*" symbol denote the default used.*

4.6.2 SME Signature/Verification Process

The SME signature/verification process begins when a subscriber publishes a certificate to a Certification Authority. Published certificates contain the subscriber's identity and public key, and are digitally signed by the Certification Authority. The Certification Authority is also responsible for safeguarding access to the subscriber's private key, which is required during the verification process.

When a subscriber signs and sends a message, the SME **Sign** process converts the message from MIME to S/MIME format. The S/MIME message format also contains the digital footprint of the subscriber's private key, so when the message is received by another user, the public key held by the Certification Authority reads and verifies the digital signature created by the private key. This process is illustrated in Figure 23.

Figure 23 Signature/Verification Process for Secure Messaging Extension (SME)

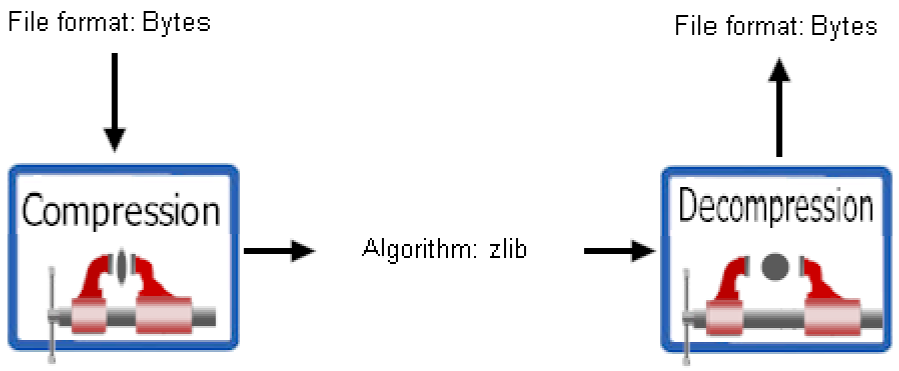

SME Sign Input Parameters
Detached Signature = True/False*
Private Key
Password
Data type = bytes
Message Format = S/MIME or PKCS7*
Sign
Alias and Password
Private Key
Certification Authority
Keystore
S/MIME Message
Alias
Public Certificate
SME Verify Input Parameters
Sender's Public Certificate
Verify

Note: *Input parameters listed with a "*" symbol denote the default used.*

4.6.3 SME Compression/Decompression Process

The SME compression process converts byte type files into PKCS#7 format using the zlib compression library, as illustrated in Figure 24.

Figure 24 Compression/Decompression Process for Secure Messaging Extension (SME)



- For more information on PKCS#7, see the *Secure Messaging Extension User's Guide*.
- For more information on the zlib compression library, visit the gzip home page at: **http://www.gzip.org**

4.7 SEF OTD Wizard

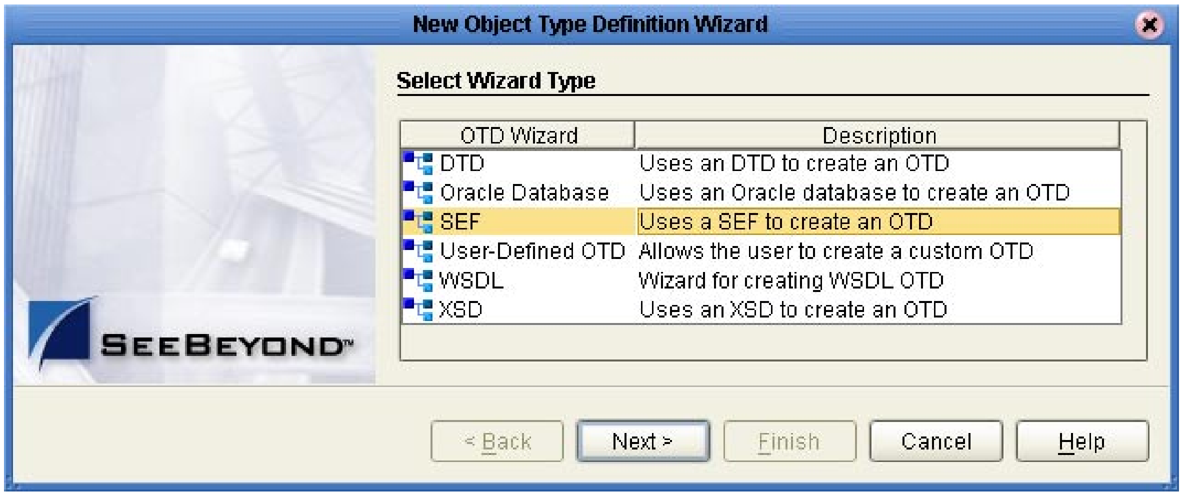
Included with eXchange is the SEF OTD Wizard. The wizard allows you to generate OTDs for ASC/X12, HIPAA, UN/EDIFACT, and related standard EDI formats using Standard Exchange Format (SEF) files—***.sef**—as input.

Note: *For detailed information on SEF and* ***.sef*** *files, including automated tools that can be used to generate* ***.sef*** *files for a variety of EDI standards: contact the Foresight Corporation:* **http://www.foresightcorp.com**

To create a SEF-based OTD

1 Right-click the project and, on the popup context menu, point at **New**, and click **Object Type Definition**. Then, in the Select Wizard Type step, click **SEF** and then click **Next**. See Figure 25.

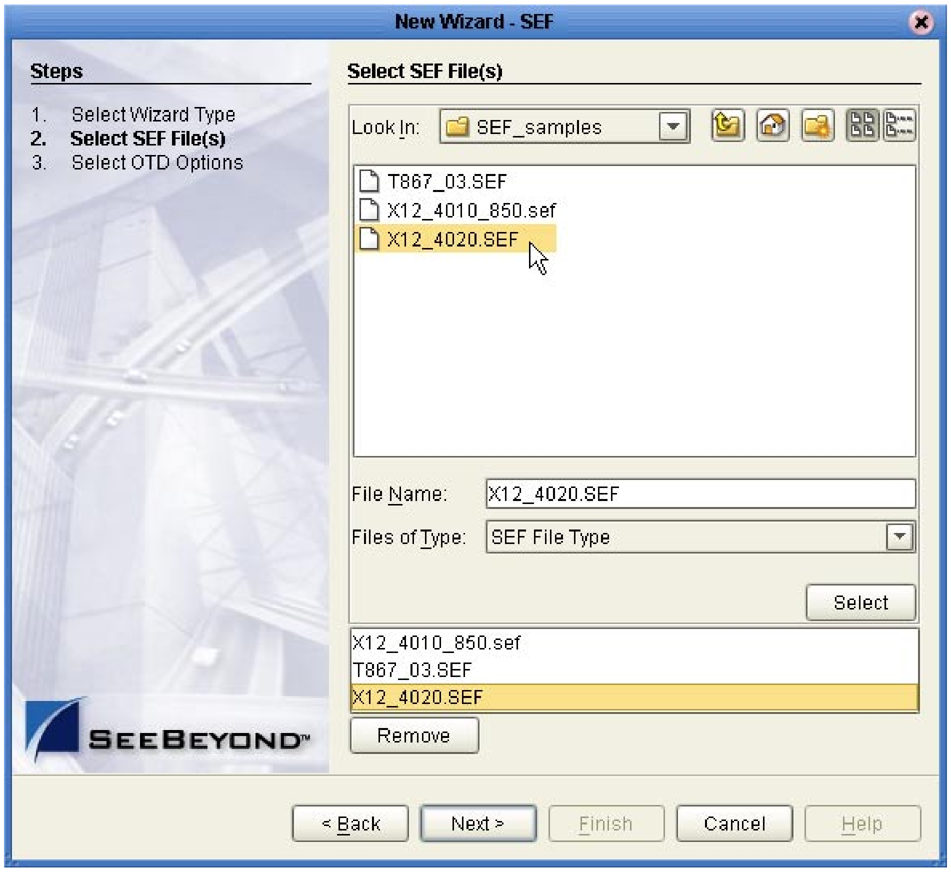
Figure 25 Selecting SEF from the New Object Type Definition Wizard



OTD Wizard	Description
DTD	Uses an DTD to create an OTD
Oracle Database	Uses an Oracle database to create an OTD
SEF	Uses a SEF to create an OTD
User-Defined OTD	Allows the user to create a custom OTD
WSDL	Wizard for creating WSDL OTD
XSD	Uses an XSD to create an OTD

2 In step 2 (**Select SEF Files**), navigate to the location(s) of the **.sef** file(s) you want to create OTDs from, select one or more **.sef** files, and then click **Next**. See Figure 26.

Figure 26 SEF Wizard: Selecting One or More SEF Files for OTD Creation



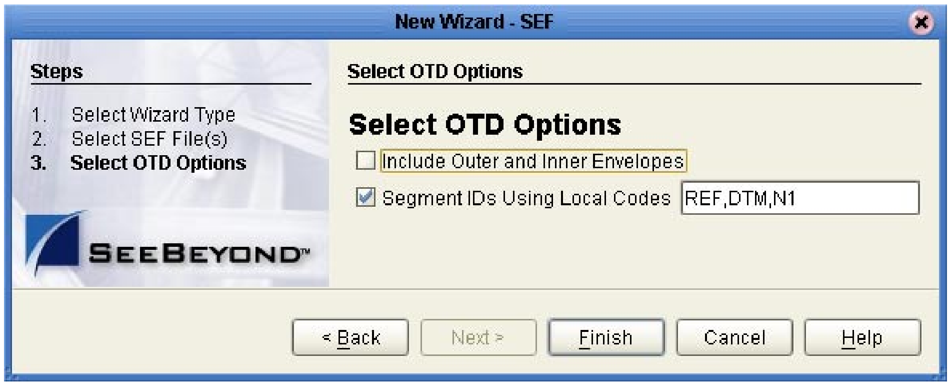
File Name: X12_4020.SEF
Files of Type: SEF File Type

3 In step 3 (**Select OTD Options**), specify:

- **Include Outer and Inner Envelopes**—When this check box is selected, the generated OTD(s) will include the outer and inner envelope segments:
  - For an X12 OTD, these segments include interchange envelope segments (ISA, IEA) and functional group envelope segments (GS, GE).
  - For an EDIFACT OTD (batch only; EDIFACT interactive messages are not supported), these segments include interchange group envelope segments (UNA, UNB, UNZ) and functional group envelope segments (UNG and UNE).
- **Segment IDs Using Local Codes**—When this check box is selected, the text entry area allows you to enter a list of segments separated by commas (such as "REF,DTM,N1"). This means the OTD runtime will use local codes specified for each of these segments to parse incoming data. Local codes are often used to distinguish adjacent segment data with same segment IDs.

An example appears in Figure 27.

Figure 27 SEF Wizard: Selecting Options for OTD Creation



Select OTD Options

Select OTD Options

☐ Include Outer and Inner Envelopes

☑ Segment IDs Using Local Codes REF,DTM,N1

4 When you have finished indicating your choices, click **Finish**.

Result: The generated OTD is displayed in the OTD Editor.

Chapter 5

Using eXchange in Enterprise Designer

This chapter provides step-by-step procedures for using the Enterprise Designer tools, editors, components, and prebuilt protocols and libraries provided by eXchange.

Process Overview

eXchange centers around the concept of a *delivery channel profile* for each trading partner relationship. A delivery channel profile (DCP) consists of:

- A set of *attribute definitions* for transport and messaging, and also, in some cases, for enveloping(=packaging). Attribute definitions define metadata, such as parameter names and types. Later, using ePM, trading parameter values will be supplied.

  The B2B protocol manager composite applications available for eXchange, such as AS2 or ebXML, come equipped with prebuilt messaging attribute definitions, while for custom B2B protocols, you set up your own. Similarly for transport: You can use either the standard transport attribute definitions supplied with eXchange (HTTP, FTP, ...) or create custom ones that you set up yourself.

- A *messaging service*, created using the *eXchange Service Designer*, that choreographs the message-exchange interactions between your enterprise and trading partners. Each messaging service is based on particular messaging attributes definition.

The DCPs—that is, the associations between messaging services and transport attribute definitions—are housed in a *B2B host*. After the B2B host is set up with all its DCPs, a connectivity map is created to connect its output (as well as the output of the Message Tracker application), to an Oracle eWay communicating with the eXchange database. Activation causes the DCPs to be stored in the eXchange database, and also creates an external server, an *eXchange Service*, in the same Environment that contains the Oracle external. If Secure Messaging Extension (SME) is installed, the eXchange Service corresponding to the B2B Host can then be configured with keystores, trust stores, and certificates. The eXchange Service is responsible for *channel management*.

For business logic, prebuilt B2B protocol processes for some standard B2B protocols are supplied with the eXchange product. In addition, the *eXchange Protocol Designer* can be used to design and configure custom B2B protocol processes that you create.

B2B protocol processes for inbound and/or outbound messages are dragged into a connectivity map, where they are represented as services. There, they are connected in usual fashion with externals (including the eXchange Service for channel management) and with other services. Activation of a corresponding Deployment Profile exposes the map's components for processing by logical hosts. As before, it also stores the DCPs in the eXchange database, making them available to eXchange Partner Manager (ePM).

In this chapter

- Steps for creating a B2B host and populating it with attribute definitions.
- Optional steps for creating and configuring custom attribute definitions.
- Steps for designing messaging services.
- Steps for creating and configuring external delivery channels.
- Optional steps for creating and configuring internal delivery channels.
- Steps for activating a B2B host environment with a Message Tracker application. (Activating a B2B host sets the stage for later activation of projects using the Channel Manager capabilities provided by the eXchange Service corresponding to the B2B host.)
- Optional steps for configuring an eXchange Service with certificates and truststores.

5.1 Setting Up a B2B Host and Its Components

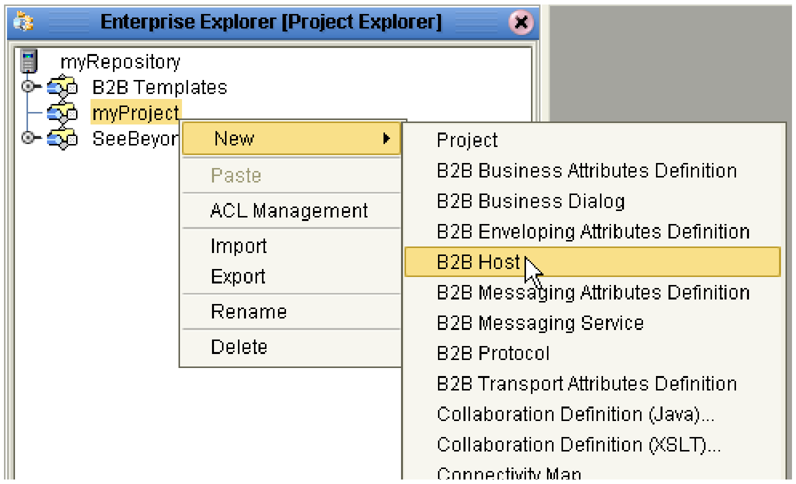
This section explains how to create a B2B host and populate its top window (Business Protocols) with attributes definitions.

The editor used for configuring B2B hosts is the **B2B Host Designer**.

To create a B2B host

1. In Enterprise Designer with the Project Explorer tab active, in the project tree, right-click the project or subproject where the B2B host will reside.
2. On the popup context menu, point at **New**, and then click **B2B Host**. See Figure 28.

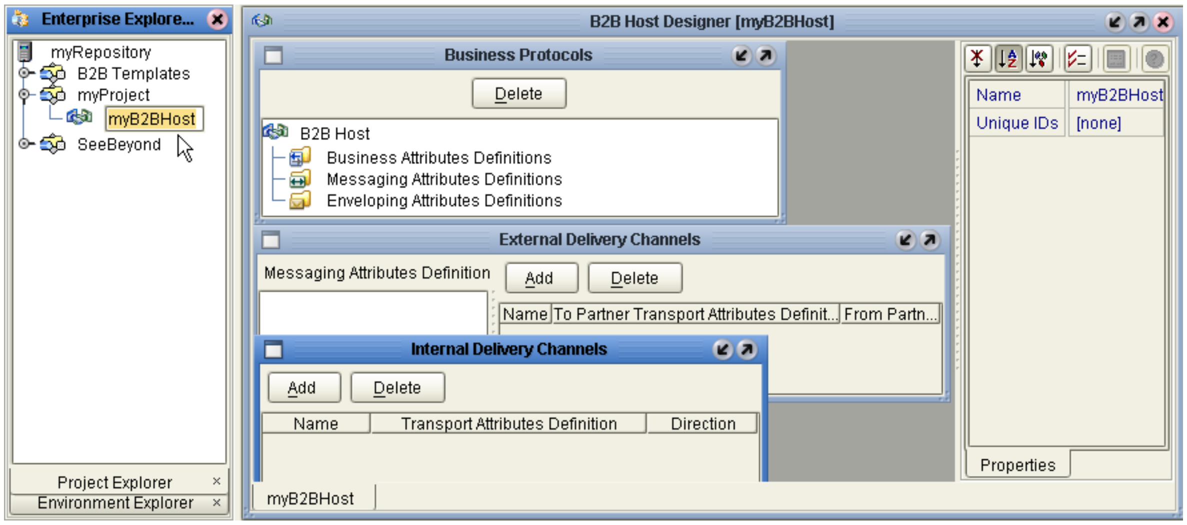
Figure 28 Creating a New B2B Host


Enterprise Explorer [Project Explorer]
myRepository
B2B Templates
myProject
SeeBeyon
New
Paste
ACL Management
Import
Export
Rename
Delete
Project
B2B Business Attributes Definition
B2B Business Dialog
B2B Enveloping Attributes Definition
B2B Host
B2B Messaging Attributes Definition
B2B Messaging Service
B2B Protocol
B2B Transport Attributes Definition
Collaboration Definition (Java)...
Collaboration Definition (XSLT)...
Connectivity Map

3. When the new component appears in the project tree, you can click-pause-click to rename it from **B2BHost1** to whatever you want.

Result: The project tree displays the new component, whose icon suggests two partners shaking hands. Also, the **B2B Host Designer** opens to display three blank windows—the Business Protocols, External Delivery Channels, and Internal Delivery Channels windows— as well as a Properties window on the far right. See Figure 29.

Figure 29 B2B Host Designer


Enterprise Explore...
myRepository
B2B Templates
myProject
myB2BHost
SeeBeyond
Project Explorer
Environment Explorer
B2B Host Designer [myB2BHost]
Business Protocols
Delete
B2B Host
Business Attributes Definitions
Messaging Attributes Definitions
Enveloping Attributes Definitions
External Delivery Channels
Messaging Attributes Definition
Add
Delete
Name
To Partner Transport Attributes Definit...
From Partn...
Internal Delivery Channels
Add
Delete
Name
Transport Attributes Definition
Direction
myB2BHost
Properties

Name	myB2BHost
Unique IDs	[none]

B2B Host Designer windows

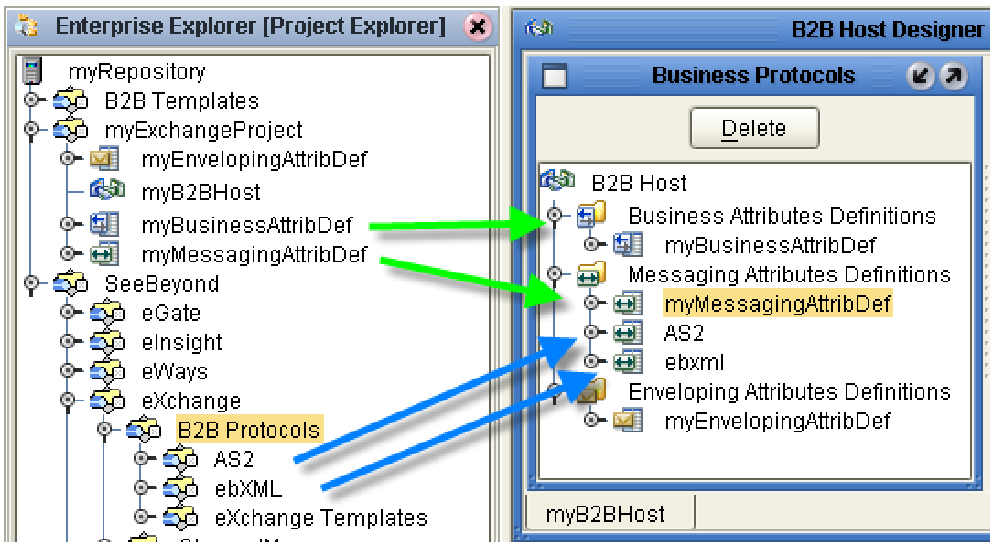
- The **Business Protocols** window displays a tree of attribute definitions and services that are exposed to the B2B host. Services are not added directly to the tree, but are instead organized by attributes definition. This window allows you drag attribute definitions into the tree, and then to drag messaging services under the corresponding attribute definition folder.
- The **External Delivery Channels** window lists these messaging attributes definitions in its left pane, while its right pane lists delivery channels defined for the item highlighted on the left. It allows you to add, delete, and modify the properties of external delivery channels.
- The **Internal Delivery Channels** window lists names, transport attributes definitions, and direction (Sender = to Internal; Receiver = from Internal). It allows you to add and delete IDCs.
- The **Properties** window, as usual, shows the properties of the currently selected component (initially, the properties of the B2B Host itself) and allows you to modify them.

To populate a B2B host with attributes definitions

- For *custom* messaging attribute definitions (ones that you have defined yourself): Open the project or subproject and drag the attributes definition into the Business Protocols window (see upper arrows in Figure 30). Repeat as needed for other custom attribute definitions.
- For *SeeBeyond-supplied* B2B protocol attribute definitions: In the project explorer tree, open the SeeBeyond > eXchange > **B2B Protocols** folder, and then:
    - *For AS2:* Open the **AS2** folder and drag the **AS2** messaging attributes definition (at the end of the list) into the B2B Host Designer canvas, in the Business Protocols window, under the Messaging Attributes Definitions node.
    - *For ebXML:* Open the **ebXML** folder and drag the **ebxml** messaging attributes definition (at the end of the list) into the B2B Host Designer canvas, in the Business Protocols window, under the Messaging Attributes Definitions node.

Result: Figure 30 shows a Business Protocols window where both custom and standard attributes definitions have been dragged into the tree.

Figure 30 Business Protocols Window



5.2 Setting Up Custom Attribute Definitions

This section explains how to create and configure custom attribute definitions:

- **Creating and Configuring Transport Attribute Definitions** on page 78
- **Creating and Configuring Messaging Attribute Definitions** on page 82

5.2.1. Creating and Configuring Transport Attribute Definitions

In general, a *protocol* is a code of behavior: a framework for interpretation and communication that is agreed upon by all parties. It prescribes rules for interacting with others who are using the same protocol.

A *transport protocol* provides a way of specifying how data is to be delivered from one system to another. For example, FTP (file transfer protocol) requires the client to specify a transfer mode (such as ASCII or binary), a target directory, a target filename or file pattern, and so forth; in eXchange, these parameters are specified by the standard *transport attributions definition* for FTP. eXchange supplies attribute definitions for the following standard transport protocols: FILE, FTP, HTTP, HTTPS, JMS, and SMTP.

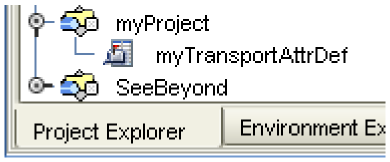
In addition to the attribute definitions for the standard transport protocols noted above, you can use ***custom transport attribute definitions*** that specify custom modifications or extensions of the standard transport protocols.

To create a custom transport attributes definition

1. In Enterprise Designer with the Project Explorer tab active, in the project tree, right-click the project or subproject where the transport attributes definition will reside.
2. On the popup context menu, point at **New**, and then click **B2B Transport Attributes Definition**.

   The project tree displays the new component, which has a icon. Its default name is **B2BTransportAttributesDefinition<*n*>**, but you can rename it however you like.

Figure 31 Custom Transport Attributes Definition



This new component is useful only by virtue of its configuration—you will need to add and define attributes that govern the nodes in the OTD that will be generated from it. Once attributes are defined, they can be exposed to eXchange Partner Management for delivery channel configuration.

To configure a custom transport attributes definition

1 In the project tree, right-click the transport attributes definition you want to modify.

Note: *If the component is locked, you must check it out before you can modify it.*

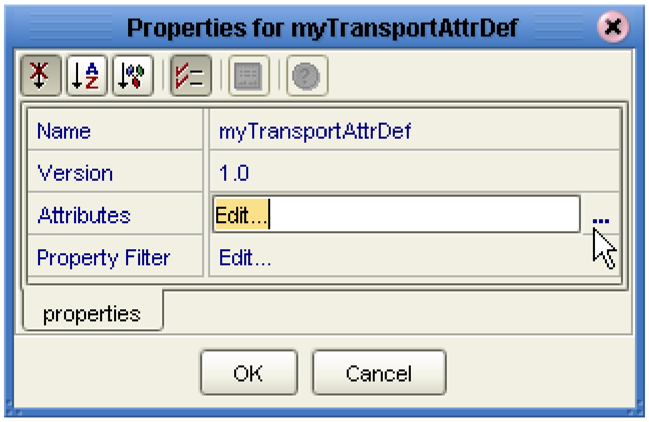
2 On the popup context menu, click **Properties**.

The properties dialog appears. See Table 30 and Figure 32.

Table 30 Properties of a Transport Attributes Definition

Property Name	Initial or Default Value	Description
Attributes	(initially null)	An XML string that holds the values displayed in the **Attributes** dialog. To open the dialog, click the ellipsis **[...]** button on the far right.
Name	(component name)	Free text; this is the component name displayed in the project tree.
Property Filter	(initially null)	An XML string that holds the values displayed in the **Property Filter** dialog. To open the dialog, click the ellipsis **[...]** button on the far right.
Version	**1.0**	Free text; reflects whatever version naming conventions you use. This is ***independent*** of the Repository-assigned versioning, which tracks check-outs/check-ins.

Figure 32 Properties for Transport Attributes Definition

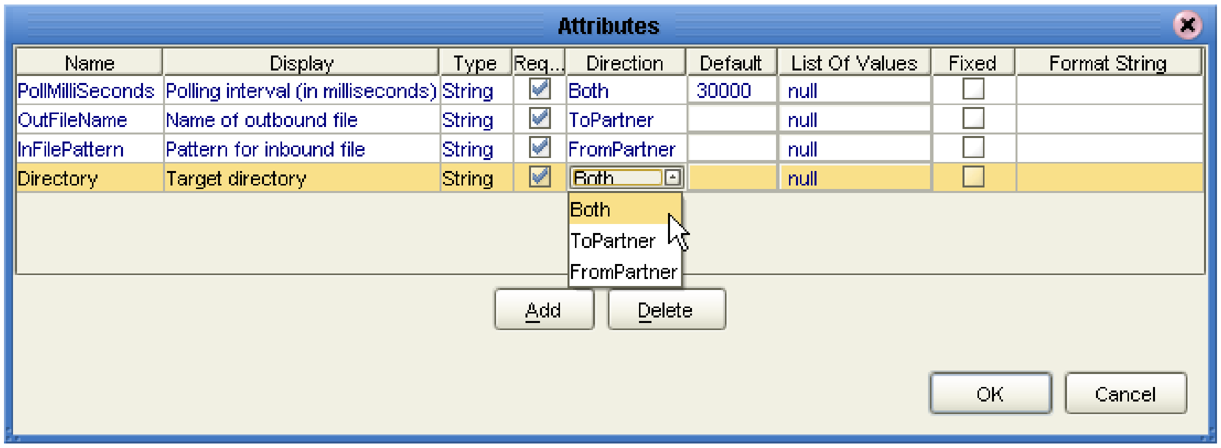


Name	myTransportAttrDef
Version	1.0
Attributes	Edit...
Property Filter	Edit...

3 To the far right of the value for Attributes, click the ellipsis [...] button.

The **Attributes** dialog appears; Figure 33 shows attributes for a sample bidirectional transport attributes definition that is a modification of the basic File TAD.

Figure 33 Custom Attribute Definitions



Name	Display	Type	Req...	Direction	Default	List Of Values	Fixed	Format String
PollMilliSeconds	Polling interval (in milliseconds)	String	✓	Both	30000	null	☐	
OutFileName	Name of outbound file	String	✓	ToPartner		null	☐	
InFilePattern	Pattern for inbound file	String	✓	FromPartner		null	☐	
Directory	Target directory	String	✓	Both		null	☐	

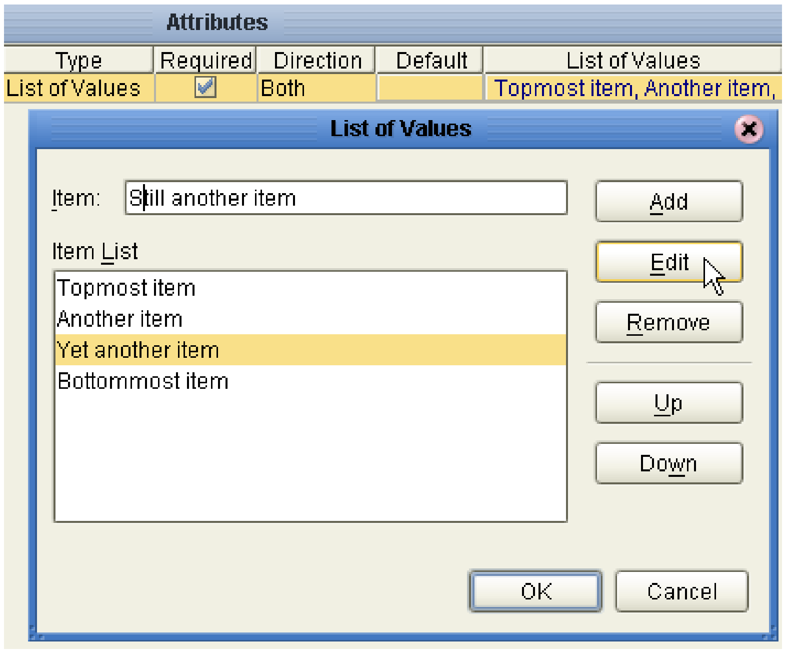
You use this dialog to create and set attributes. These values govern the appearance and behavior of the parameters displayed in eXchange Partner Management (ePM) when configuring external delivery channels for a trading partner profile, in the ToPartner Transport and FromPartner Transport subtabs.

4 Click the **Add** button as many times as needed and then, for each row created:

- Change **Name** to a meaningful node name for the OTD you will generate.
- Change **Display** to the text you want to display as a prompt or label for the parameter in ePM.
- For **Type**, select the data type for this attribute. The default, **String**, allows the ePM user to enter any character data; **Password** also accepts any ePM input, and masks the input; **Integer** accepts positive or negative whole numbers only; **Number** extends this to also accept decimal numbers (floating-point numbers); **Boolean** requires the ePM user to make a yes-or-no choice; **List of Values** presents the ePM user with a drop-down list restricted to the items you have set up (see below); and **DateTime** prompts the ePM user to supply a date and/or time value, based on the formatting you provide (see Format String, below).
- For **Required**, select or clear the box according to whether you want the parameter to be a required or optional entry. (In the ePM GUI, parameters that have been designated as required are flagged with a red asterisk.)
- For **Direction**, choose **ToPartner**, **FromPartner**, or **Both** according to whether you want the parameter to appear with the ToPartner parameters, FromPartner parameters, or both.
- For **Default**, you can optionally enter a default value that will appear in ePM before the user enters data or makes a selection. This is the value that will be used if it is not overridden by the ePM user.

- For **List of Values**, which is available only for an attribute whose data type is "List of Values" (see Type, above), double-click on the ellipsis [**...**] button to the far right and use the **List of Values** dialog box to add entries to the drop-down list that will be seen by the end user. Clicking Add appends a new item to the end of the list; Edit modifies the currently selected item (see Figure 34); Up and Down move it higher or lower in the list.

Figure 34 Adding a List of Values to an Attributes Definition



Type	Required	Direction	Default	List of Values
List of Values	✓	Both		Topmost item, Another item,

Item: Still another item

Item List:
Topmost item
Another item
Yet another item
Bottommost item

- **Format String** allows you to use special characters as shorthand for certain often-used information; for example, **%f** is the working filename, **%M** is the current month, **%d** the current day, and so forth. For more information, see the *Batch eWay Intelligent Adapter User's Guide*; the chapter on understanding OTDs has a section on using special characters.

5 When you have finished adding and modifying attributes, click **OK**.

6 For **Property Filter**: If you want to change the packaging filters for this custom protocol (by default, both encryption truststore and signing certificates are filtered for both FromPartner bindings and ToPartner bindings), click the ellipsis [**...**] button at the far right and then, in the **Property Filter** dialog, clear the checkboxes for the filters you want disabled. When you have finished, click **OK**.

7 Click **OK** to close the properties dialog.

After you have completed these steps, the transport attributes definition will appear as a choice in the drop-down list when you configure the delivery channels of your B2B host.

5.2.2. Creating and Configuring Messaging Attribute Definitions

A *messaging protocol* provides a way of specifying how data is bundled and unbundled—for example, it is at this level that encryption, acknowledgment, and nonrepudiation are addressed.

In addition to the attribute definitions for the standard B2B protocols noted above, you can use ***messaging attribute definitions*** to create modifications or extensions of the standard messaging attributes definitions. These variants are called *custom* messaging attributes definitions.

To create a messaging attribute definition

1 In Enterprise Designer with the Project Explorer tab active, in the project tree, right-click the project or subproject where the messaging attribute definition will reside.

2 On the popup context menu, point at **New**, and then click **B2B Messaging Attributes Definition**.

3 In **Create B2B Messaging Attributes Definition**, enter a name and click **OK**.

The project tree displays the new component, which has a icon.

This new component is useful only by virtue of its configuration—you will need to add name/value pairs for the parameters you want it to define.

To configure a messaging attribute definition

1 In Enterprise Designer, with the Project Explorer tab active, right-click the messaging attribute definition you want to modify.

Note: *If the component is locked, you must check it out before you can modify it.*

2 On the popup context menu, click **Properties**. See Table 31.

Table 31 Properties for Messaging Attribute Definition

Property Name	Initial or Default Value	Description
Attributes	(initially null)	An XML string that holds the values displayed in the **Attribute** dialog. To open the dialog, click the ellipsis **[...]** button on the far right.
Name	(component name)	Free text; this is the component name displayed in the project tree.
Version	**1.0**	Free text; reflects whatever version naming conventions you use. This is ***independent*** of the Repository-assigned versioning, which tracks check-outs/check-ins.

3 To the far right of the value for **Attributes**, click the ellipsis [...] button.

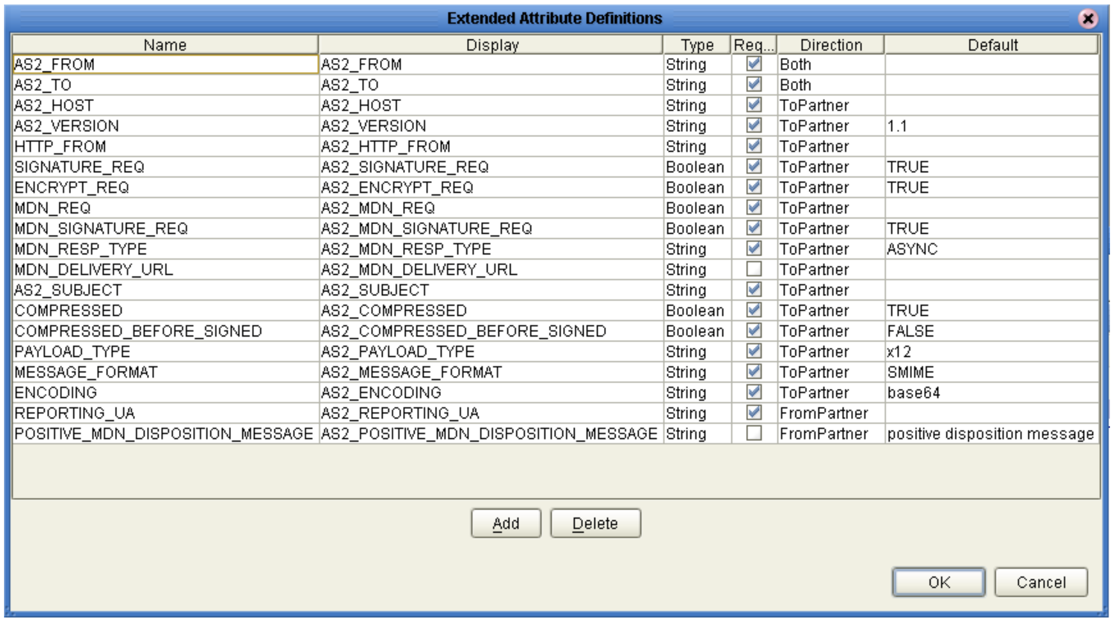
The **Protocol Attribute Definitions** dialog appears. You use this dialog to create and set the attributes. These values govern the appearance and behavior of the parameters displayed in ePM when configuring external delivery channels for a trading partner (the ToPartner Transport and FromPartner Transport subtabs).

4 Click the **Add** button as many times as needed and then, for each row created:

- Change **Name** to a meaningful node name for the OTD you will generate.
- Change **Display** to the text you want to display as a prompt or label for the parameter in ePM.
- For **Type**, select the data type for this attribute. The default, **String**, allows the ePM user to enter any character data; **Password** also accepts any ePM input, and masks the input; **Integer** accepts positive or negative whole numbers only; **Number** extends this to also accept decimal numbers (floating-point numbers); **Boolean** requires the ePM user to make a yes-or-no choice; **List of Values** presents the ePM user with a drop-down list restricted to the items you have set up (see below); and **DateTime** prompts the ePM user to supply a date and/or time value, based on the formatting you provide (see Format String, below).
- For **Required**, select or clear the box according to whether you want the parameter to be a required or optional entry. (In the ePM GUI, parameters that have been designated as required are flagged with a red asterisk.)
- For **Direction**, choose **ToPartner**, **FromPartner**, or **Both** according to whether you want the parameter to appear with the ToPartner parameters, FromPartner parameters, or both.
- For **Default**, you can optionally enter a default value that will appear in ePM before the user enters data or makes a selection. This is the value that will be used if it is not overridden by the ePM user.
- For **List of Values**, which is available only for an attribute whose data type is “List of Values” (see Type, above), double-click on the ellipsis [**...**] button to the far right and use the **List of Values** dialog box to add entries to the drop-down list that will be seen by the end user. Clicking Add appends a new item to the end of the list; Edit modifies the currently selected item (see Figure 34); Up and Down move it higher or lower in the list.
- **Format String** allows you to use special characters as shorthand for certain often-used information; for example, **%f** is the working filename, **%M** is the current month, **%d** the current day, and so forth. For more information, see the *Batch eWay Intelligent Adapter User’s Guide*; the chapter on understanding OTDs has a section on using special characters.

You previously saw an example of a custom transport attributes definition (in **Figure 33 on page 80**); as a different example, the messaging attributes of a standard B2B protocol (in this case, AS2 version 1.1) are shown in Figure 35.

Figure 35 Messaging Attribute Definitions for AS2 Version 1.1



Name	Display	Type	Req...	Direction	Default
AS2_FROM	AS2_FROM	String	✓	Both	
AS2_TO	AS2_TO	String	✓	Both	
AS2_HOST	AS2_HOST	String	✓	ToPartner	
AS2_VERSION	AS2_VERSION	String	✓	ToPartner	1.1
HTTP_FROM	AS2_HTTP_FROM	String	✓	ToPartner	
SIGNATURE_REQ	AS2_SIGNATURE_REQ	Boolean	✓	ToPartner	TRUE
ENCRYPT_REQ	AS2_ENCRYPT_REQ	Boolean	✓	ToPartner	TRUE
MDN_REQ	AS2_MDN_REQ	Boolean	✓	ToPartner	
MDN_SIGNATURE_REQ	AS2_MDN_SIGNATURE_REQ	Boolean	✓	ToPartner	TRUE
MDN_RESP_TYPE	AS2_MDN_RESP_TYPE	String	✓	ToPartner	ASYNC
MDN_DELIVERY_URL	AS2_MDN_DELIVERY_URL	String	☐	ToPartner	
AS2_SUBJECT	AS2_SUBJECT	String	✓	ToPartner	
COMPRESSED	AS2_COMPRESSED	Boolean	✓	ToPartner	TRUE
COMPRESSED_BEFORE_SIGNED	AS2_COMPRESSED_BEFORE_SIGNED	Boolean	✓	ToPartner	FALSE
PAYLOAD_TYPE	AS2_PAYLOAD_TYPE	String	✓	ToPartner	x12
MESSAGE_FORMAT	AS2_MESSAGE_FORMAT	String	✓	ToPartner	SMIME
ENCODING	AS2_ENCODING	String	✓	ToPartner	base64
REPORTING_UA	AS2_REPORTING_UA	String	✓	FromPartner	
POSITIVE_MDN_DISPOSITION_MESSAGE	AS2_POSITIVE_MDN_DISPOSITION_MESSAGE	String	☐	FromPartner	positive disposition message

5 When you have finished adding and modifying attributes, click **OK**.

6 Click **OK** to close the properties dialog.

The messaging attribute definition can now be used to generate an OTD.

After you have completed these steps, the messaging attribute definition will appear as a choice in the drop-down list of messaging attributes definitions when you configure the delivery channels of your B2B host.

5.3 Setting Up Messaging Services

A messaging service is a project-level (logical) component that sets forth rules of exchange between an enterprise and its trading partners.

The section explains how to create, populate, configure, and compile a messaging service, using the **eXchange Service Designer** editor. These steps must be followed to allow the B2B protocol to be used by a B2B host.

To create and configure a messaging service

1 In Enterprise Designer with the Project Explorer tab active, in the project tree, right-click the project or subproject where the messaging service will reside.

2 On the popup context menu, point at **New**, and then click **messaging service**.

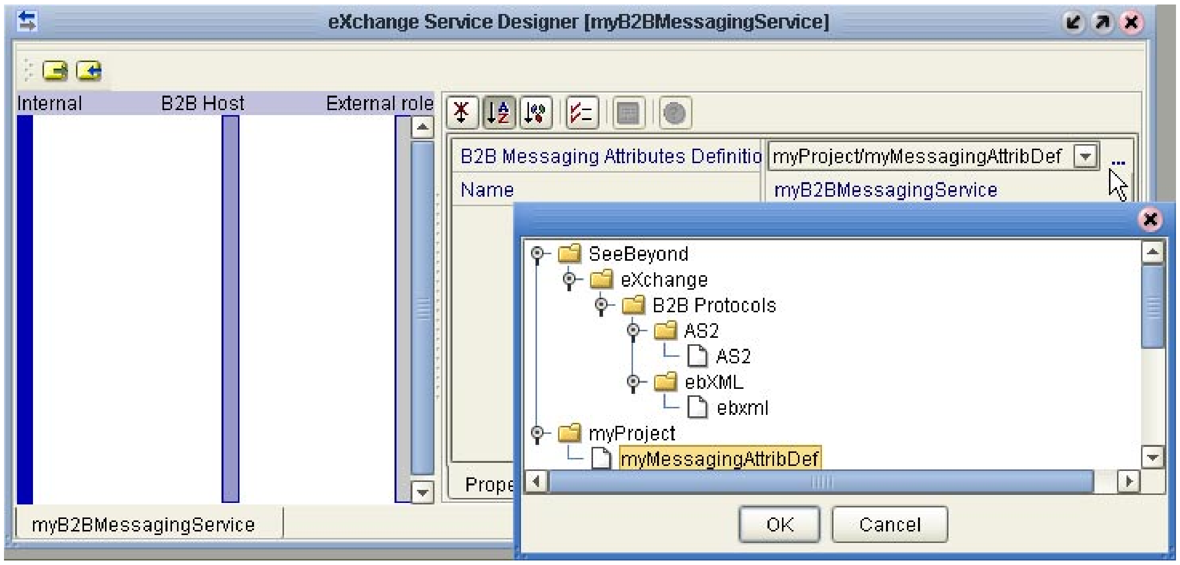
The project tree displays the new component, whose icon suggests bidirectional passthrough communication. By default, it is named **B2B Messaging Service<*n*>**, but you can rename it as you like.

The eXchange Service Designer opens to display a blank canvas.

3 For B2B Messaging Attributes Definition, choose an entry in the drop-down list. This list will include all enveloping attributes definitions supplied by SeeBeyond (such as for AS2 and ebXML, if you have installed them), and will also show you any custom messaging attribute definitions.

Alternatively, you can click its ellipsis [...] button to open a new window whose tree contains messaging services you can pick; see Figure 37.

Figure 36 Using eXchange Service Designer to Choose a Messaging Service



B2B Messaging Attributes Definitio: myProject/myMessagingAttribDef
Name: myB2BMessagingService

- SeeBeyond
  - eXchange
    - B2B Protocols
      - AS2
        - AS2
      - ebXML
        - ebxml
- myProject
  - myMessagingAttribDef

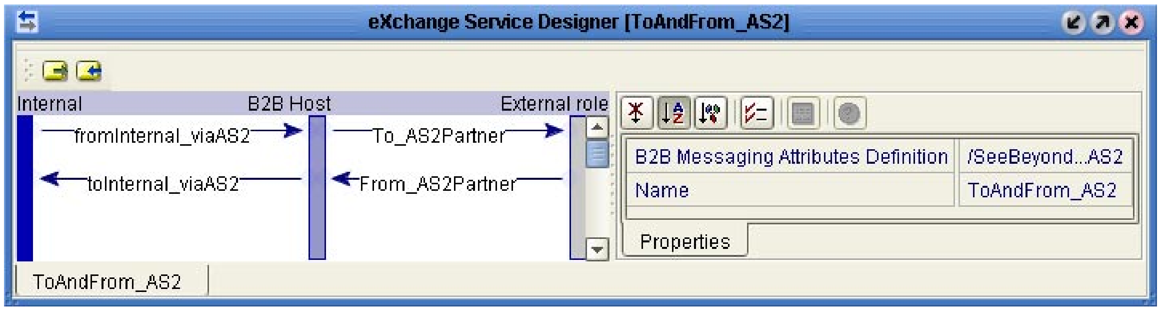
This new component still needs to be populated—you will need to add actions that describe the interchange of messages to and from the trading partner.

4 In eXchange Service Designer, use the icons in upper left to do at least one of the following:

- Click **Message_In** (the right-pointing arrow icon) to add an action that originates in the enterprise and passes through the B2B host to the trading partner. Then, double-click, the label on each of the two right-pointing arrows, editing the names to whatever you want.
- Click **Message_Out** (the left-pointing arrow icon) to add an action that originates in the trading partner and passes through the B2B host to the enterprise. Then, double-click the label on each of the two left-pointing arrows, editing the names to whatever you want.

5 Optionally, add other activities as needed. See Figure 37.

Figure 37 Messaging Service With Outbound and Inbound Activities



B2B Messaging Attributes Definition	/SeeBeyond...AS2
Name	ToAndFrom_AS2

Depending on the complexity of interaction between the enterprise and the trading partner, a messaging service might have only one action, or five fromInternal-toPartner actions with only one fromPartner-toInternal reply, or many paired exchanges. The characteristic quality of a messaging service is that each action passes through the B2B host; there is no separate back-and-forth interactive communication between just the B2B host and either side.

Although the actions in a messaging service can be referenced, they are not web service operations, and are therefore ineligible for dragging onto canvases. These are also the actions that appear at the lowest level of the tree in eXchange Partner Manager (ePM).

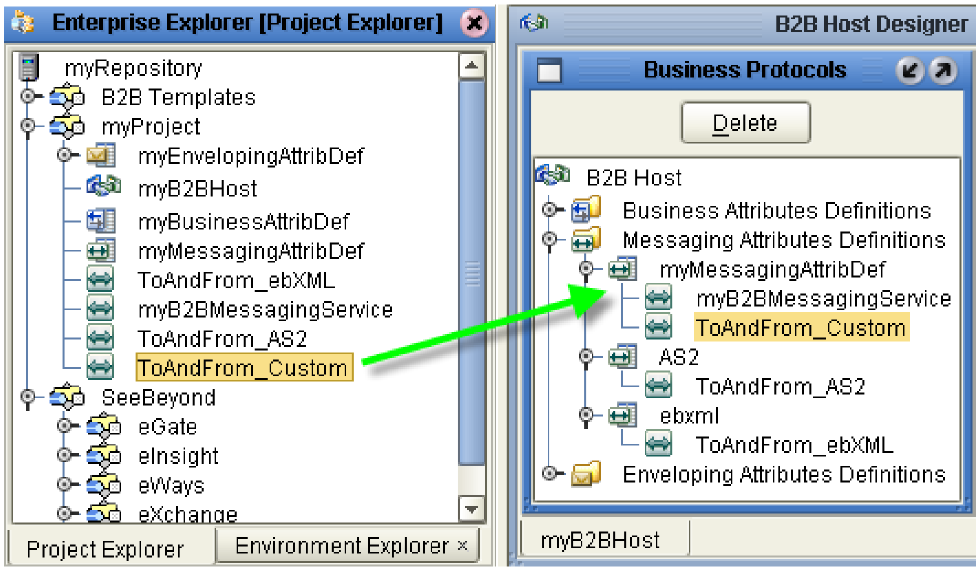
To add messaging services to a B2B host, organized by messaging attributes definition

Before you begin: The B2B Host Designer must be open.

1. Into the **Business Protocols** window, under the **Messaging Attributes Definition** folder, drag in a messaging attributes definition from the project tree. (If necessary, see **"To populate a B2B host with attributes definitions" on page 77**).

2. Under this messaging attributes definition, drag in a messaging service whose properties reference it. (The GUI does not allow you to drag in a messaging service that references a different messaging attributes definition, or none.) See Figure 38.

Figure 38 Populating the Business Protocols Tree with Message Services



The messaging service appears in the Business Protocols tree, subordinate to the messaging attributes definition it references.

3 Repeat the previous step as needed, or repeat both previous steps.

4 If you need to delete a leaf or a branch from the tree in the Business Protocols window, highlight it and click **Delete**.

5.4 Setting Up External Delivery Channels

To add external delivery channels to a B2B host

Before you begin: The External Delivery Channels window must already display at least one item in its Messaging Attributes Definitions pane; if it does not, see the previous procedure (for adding messaging services to the Business Protocols window).

1 In the **External Delivery Channels** window, in the Messaging Attributes Definitions pane, click the messaging attributes definition for which you want to add an external delivery channel.

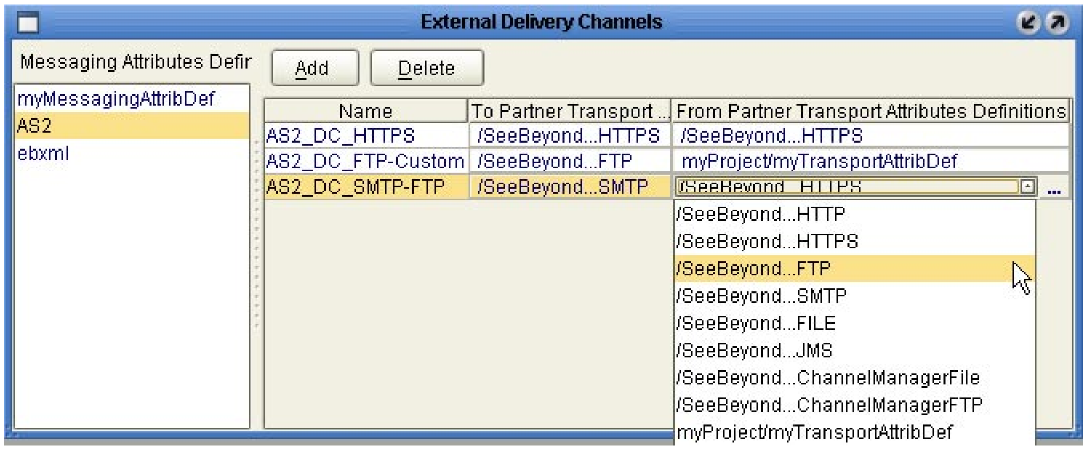
2 Click **Add**. When the new external delivery channel appears on the right, rename it.

3 In the **ToPartner Transport Attributes Definition** list, choose from the list of standard and custom transport attributes definitions displayed. This specifies transport attributes for messages outbound from the B2B host to the trading partner on this channel.

4 In the **FromPartner Transport Attributes Definition** list, choose from the list of standard and custom transport attributes definitions displayed. This specifies transport attributes for messages inbound on this channel from trading partner to B2B host.

5 Repeat the last three steps as needed, or repeat all previous steps. See Figure 39.

Figure 39 External Delivery Channels Populated and Configured

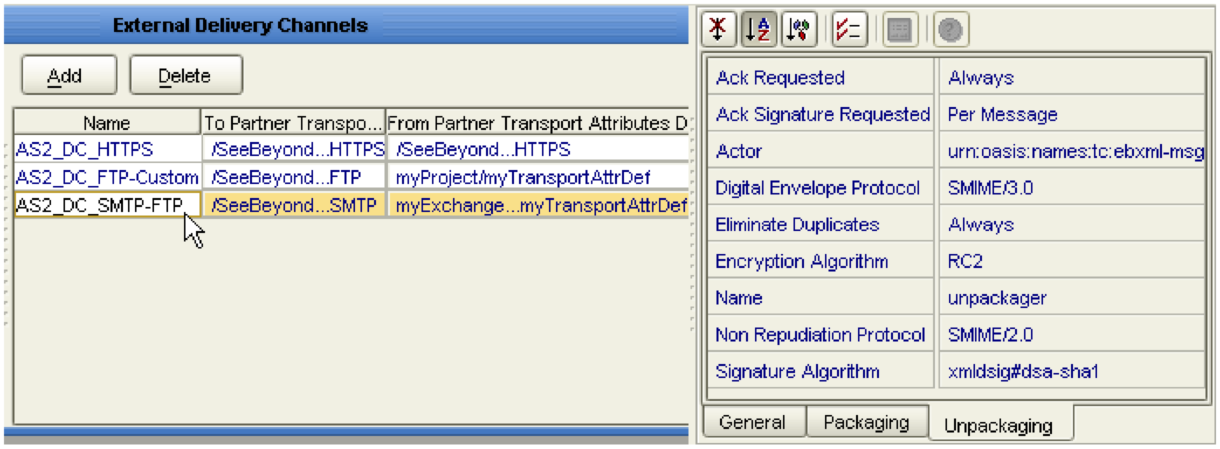

External Delivery Channels
Messaging Attributes Defir
myMessagingAttribDef
AS2
ebxml
Add
Delete
/SeeBeyond...HTTP
/SeeBeyond...HTTPS
/SeeBeyond...FTP
/SeeBeyond...SMTP
/SeeBeyond...FILE
/SeeBeyond...JMS
/SeeBeyond...ChannelManagerFile
/SeeBeyond...ChannelManagerFTP
myProject/myTransportAttribDef

Name	To Partner Transport ...	From Partner Transport Attributes Definitions
AS2_DC_HTTPS	/SeeBeyond...HTTPS	/SeeBeyond...HTTPS
AS2_DC_FTP-Custom	/SeeBeyond...FTP	myProject/myTransportAttribDef
AS2_DC_SMTP-FTP	/SeeBeyond...SMTP	/SeeBeyond...HTTPS

Note: *From the External Delivery Channels window, you can view or edit the MIME and Simple Parts properties of a particular messaging attributes definition, by clicking its name in the far left column to open its Properties sheet on the right.*

6 For any external delivery channel whose default properties you want to override, click the channel name to highlight it, opening its properties sheet on the right. See Figure 39.

Figure 40 Properties Sheet for External Delivery Channel

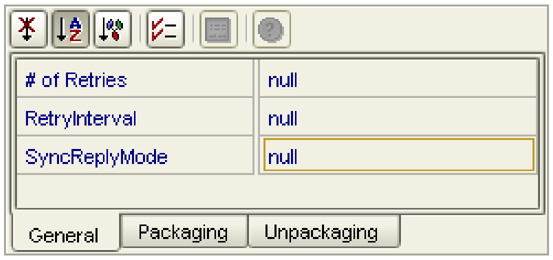


Name	To Partner Transpo...	From Partner Transport Attributes D
AS2_DC_HTTPS	/SeeBeyond...HTTPS	/SeeBeyond...HTTPS
AS2_DC_FTP-Custom	/SeeBeyond...FTP	myProject/myTransportAttrDef
AS2_DC_SMTP-FTP	/SeeBeyond...SMTP	myExchange...myTransportAttrDef

Ack Requested	Always
Ack Signature Requested	Per Message
Actor	urn:oasis:names:tc:ebxml-msg
Digital Envelope Protocol	SMIME/3.0
Eliminate Duplicates	Always
Encryption Algorithm	RC2
Name	unpackager
Non Repudiation Protocol	SMIME/2.0
Signature Algorithm	xmldsig#dsa-sha1

7 Within the property sheet, and navigate to the properties you want to modify. For general properties, see Figure 41 and Table 32; for packaging(=enveloping) and unpackaging(=de-enveloping) properties, see Figure 42 and **Table 33 on page 89**.

Figure 41 Default Property Settings for External Delivery Channels — General

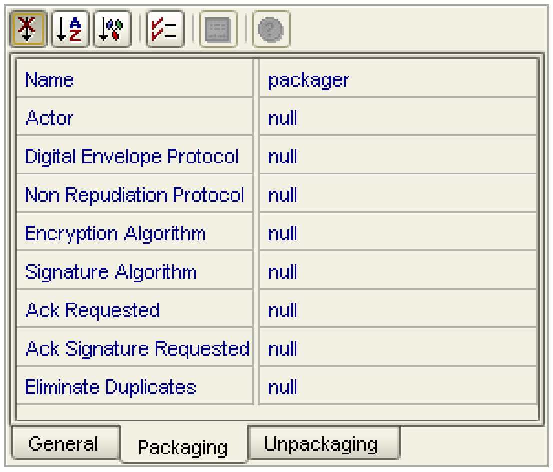


# of Retries	null
RetryInterval	null
SyncReplyMode	null

Table 32 Properties for External Delivery Channels — General

Property Name	Initial or Default Value	Description
# of retries	**null**	Number of times to retry a send of the message.
Retry interval	**null** (initially)	Interval to use for retrying a message send.
SyncReplyMode	**null** (initially)	The mode (synchronous or asynchronous) to be used for messaging.

Figure 42 Default Property Settings for External Delivery Channels — Packaging



Name	packager
Actor	null
Digital Envelope Protocol	null
Non Repudiation Protocol	null
Encryption Algorithm	null
Signature Algorithm	null
Ack Requested	null
Ack Signature Requested	null
Eliminate Duplicates	null

Table 33 Properties for External Delivery Channels — Packaging/Unpackaging

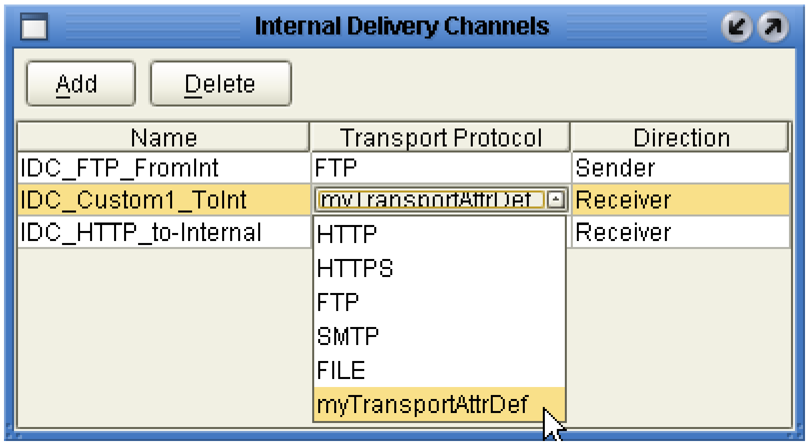
Property Name	Initial or Default Value	Description
Name	(component name)	Free text; this is the component name displayed in the project tree.
Actor	**null** (initially)	Namespace identifying the actor, such as urn:oasis:names:tc:ebxml-msg:actor:nextMSH
Digital Envelope Protocol	**null** (initially)	Digital envelope protocol and version to be used, of: **SMIME/2.0**, **SMIME/3.0**, or **DSIG/1.0**.
Nonrepudiation Protocol	**null** (initially)	Nonrepudiation protocol and version to be used, of: **SMIME/2.0**, **SMIME/3.0**, or **DSIG/1.0**.
Encryption Algorithm	**null** (initially)	Encryption algorithm to be used, of: **RC2** or **DES3**.
Signature Algorithm	**null** (initially)	Encryption algorithm to be used. If set, must be **xmldsig#dsa-sha1**.
Ack Requested	**null** (initially)	Specifies circumstances in which a message acknowledgment is requested (such as TA1), of: **Always**, **Never**, or **Per Message**.
Ack Signature Requested	**null** (initially)	Specifies circumstances in which a signature is required on the message acknowledgment, of: **Always**, **Never**, or **Per Message**.
Eliminate Duplicates?	**null** (initially)	Specifies circumstances in which duplicate messages are to be eliminated, of: **Always**, **Never**, or **Per Message**.

5.5 Setting Up Internal Delivery Channels

To add internal delivery channels to a B2B host

1. In the B2B Host Designer, in the **Internal Delivery Channels** window, click **Add** and then rename the new internal delivery channel to something meaningful.
2. In the **Transport Attributes Definition** list, choose from the list of standard and custom transport attributes definitions displayed.
3. In the **Direction** list, choose a designation of the role played by the Internal system:
   - **Sender** designates this internal delivery channel as coming from Internal.
   - **Receiver** designates this internal delivery channel as going to Internal.
4. As needed, repeat previous steps to add and configure internal delivery channels.
   Result: See Figure 43.

Figure 43 Internal Delivery Channels Populated and Configured



Name	Transport Protocol	Direction
IDC_FTP_FromInt	FTP	Sender
IDC_Custom1_ToInt	myTransportAttrDef	Receiver
IDC_HTTP_to-Internal		Receiver

5.6 Activating a B2B Host

The B2B host plays a dual role: It functions both as an *object*—that is, a project-level (logical) component in the project tree that can be dragged into a Connectivity Map—, and also as a *server*—that is, an environment (physical) component. As a server, it hosts Channel Managers and is exposed to the eXchange Partner Management GUI.

Unlike most other environment components, it is not offered as a template in the environment to be statically configured. Instead, since it has a dynamic dependency on the Oracle database that persists eXchange trading partner information (and message tracking information, if used), it must be activated for a particular environment. This section provides the steps required for activation: creating an environment, creating a map containing a B2B host, and activating a deployment profile that binds the map's components to the environment.

When a B2B host named *myHost123* is activated, the name of the new external added to the environment is *myHost123 eXchange Service*.

5.6.1. Creating an Environment

These steps create the minimal environment required to activate a B2B host.

To create and populate the environment prior to B2B host activation

1 In Enterprise Designer with the **Environment Explorer** tab active, right-click the repository and, on the popup context menu, click **New Environment**.

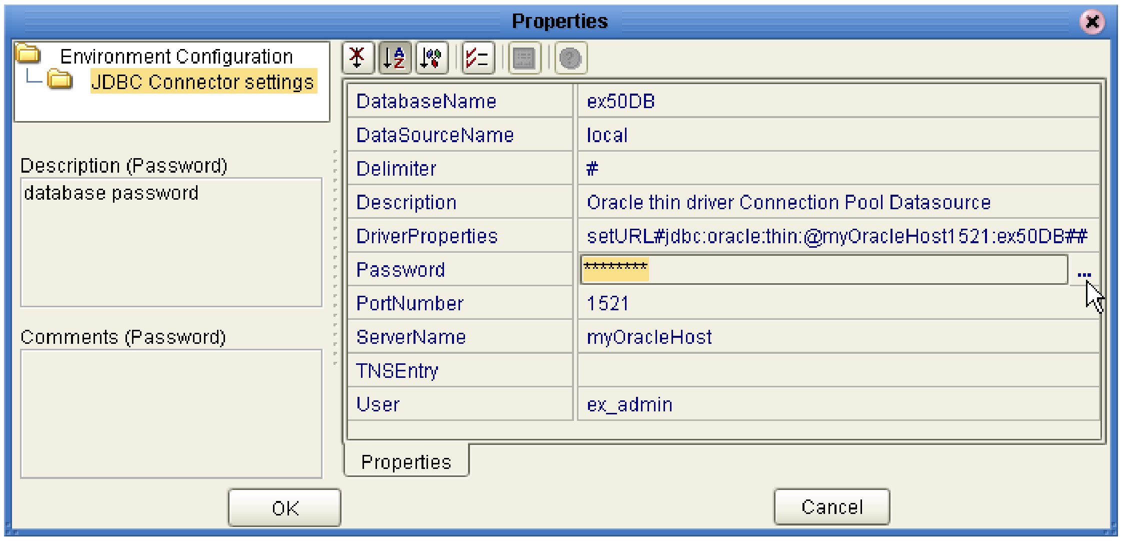
The explorer tree displays a new environment, and the Environment Editor opens. Optionally, you can rename the environment to something meaningful.

2 In the environment explorer tree, right-click the new environment and, on the popup context menu, click **New Oracle External System**.

3 In the **Create an External System** dialog, enter a meaningful name, set the system type to **Outbound Oracle eWay**, and then click **OK**.

4 Configure the Oracle external with the values for your eXchange database instance. For details on configuring the Oracle external, see the Chapter 3 of the *Oracle eWay User's Guide*. For sample settings typical of an eXchange database, see Figure 44.

Figure 44 Environment Configuration for Oracle External System



Description (Password)
database password

Comments (Password)

DatabaseName	ex50DB
DataSourceName	local
Delimiter	#
Description	Oracle thin driver Connection Pool Datasource
DriverProperties	setURL#jdbc:oracle:thin:@myOracleHost1521:ex50DB##
Password	********
PortNumber	1521
ServerName	myOracleHost
TNSEntry	
User	ex_admin

5 In the environment explorer tree, right-click the new environment again and, on the popup context menu, click **New Logical Host**.

The explorer tree and editor canvas display the new logical host. Optionally, you can use the tree to rename the logical host to something meaningful.

6 In the environment explorer tree, right-click the new logical host and, on the popup context menu, click **New SeeBeyond Integration Server**.

The explorer tree displays the new integration server, and the canvas displays it inside the logical host. Optionally, you can rename it to something meaningful.

7 If appropriate, right-click the integration server, click **Properties**, and configure its parameters as needed for use at your site.

Once you finish these last steps, the environment now has all you need to deploy a B2B host: An integration server and an Oracle external system. However, you still need to create a map that links the B2B host to an Oracle eWay.

5.6.2. Creating a Map with a B2B Host and a Message Tracker

These steps create a map that allows you to activate a B2B host with a Message Tracker.

To create and populate the map prior to B2B host deployment

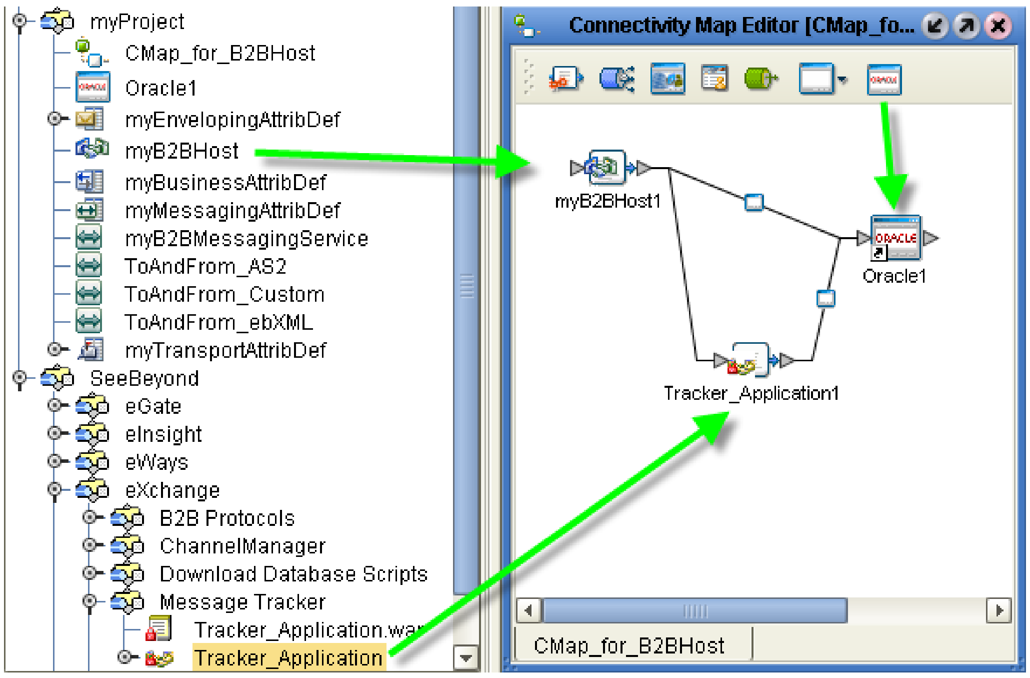
Before you begin: Your project must already contain a B2B host with at least one validly configured external delivery channel; see **To add external delivery channels to a B2B host** on page 87.

1. With the **Project Explorer** tab active, in the project tree, right-click the project and, on the popup context menu, point at click **New** and click **Connectivity Map**.

   The project tree displays a new map, and the Connectivity Map Editor opens. Optionally, you can rename the map to something meaningful.

2. In the toolbar along the top of the canvas, click the **External Applications** tool and, from the drop-down list, select the checkbox for **Oracle External Application**.
3. Drag an Oracle external onto the canvas, towards the right side.
4. From the project tree, drag your B2B host onto the upper left side of the canvas.
5. In the project tree, open the **SeeBeyond > eXchange >Message Tracker** folder and drag its **Tracker_Application** B2B protocol onto the lower center of the canvas.
6. Rename components to something meaningful, and connect as shown in Figure 45.

Figure 45 Map Showing B2B Host and Message Tracker to be Deployed



7. Configure both eWays, designating them outbound to external Oracle applications.

Now that the map is populated and configured, the B2B host is ready to be activated.

5.6.3. Activating the B2B Host

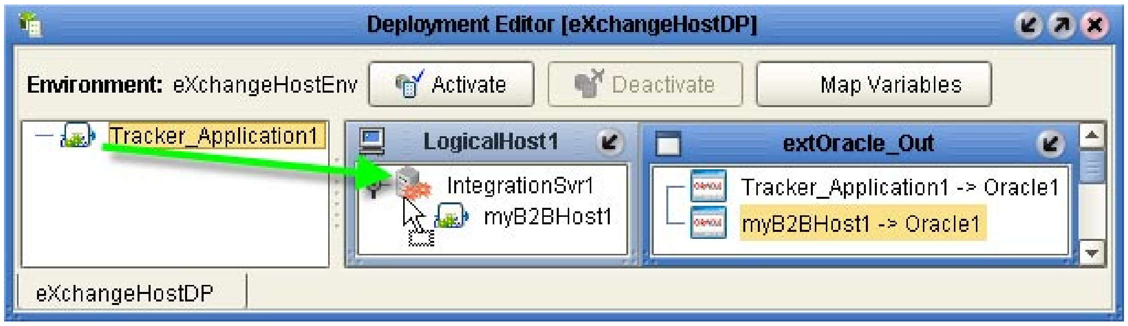
These steps create a deployment profile—a binding of particular logical components to an environment—and then activate it. Project activation always makes components available *to* external servers (such as the SeeBeyond integration server and Oracle), but activation of a B2B host makes it available ***as*** an external server. Activation also exposes the B2B host and Message Tracker application so they can communicate with the web-based eXchange GUIs: eXchange Partner Management (ePM) and Message Tracking.

To create, configure, and activate the deployment profile for the B2B host

Before you begin: Your project must already contain a validly configured B2B host used in a map that connects it, via an outbound eWay, with a configured Oracle external.

1. With the Project Explorer tab active, in the project tree, right-click the project and, on the popup context menu, point at click **New** and click **Deployment Profile**. The Deployment Editor opens a canvas whose left pane shows all components on the map and whose right pane shows all external servers in the environment.
2. One by one, drag both Oracle eWays to the Oracle server. Then, one by one, drag the B2B host and the tracker application to the integration server. See Figure 46.

Figure 46 Deployment Profile Showing B2B Host Being Assigned to Integration Server

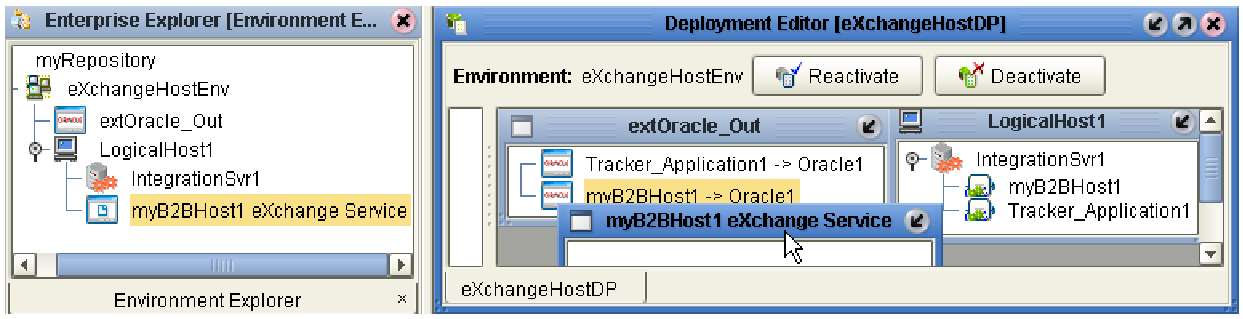


Tip: *If you cannot drop the eWays onto the Oracle server, ensure your Oracle service is running (for example: Settings > Control Panel > Administrative Tools > Services).*

3. After all components have been assigned to external servers, click **Activate**.

Upon successful activation, the Environment Editor and Deployment Editor are updated to show a new *eXchange service* associated with this B2B host. See Figure 47.

Figure 47 eXchange Service in Environment and Deployment Editor



Once it is activated, the eXchange service is capable of hosting Channel Managers, and its corresponding B2B host is available to Enterprise Manager facilities that perform B2B protocol monitoring, trading partner management, and message tracking.

To associate security information with an eXchange Service's delivery channels

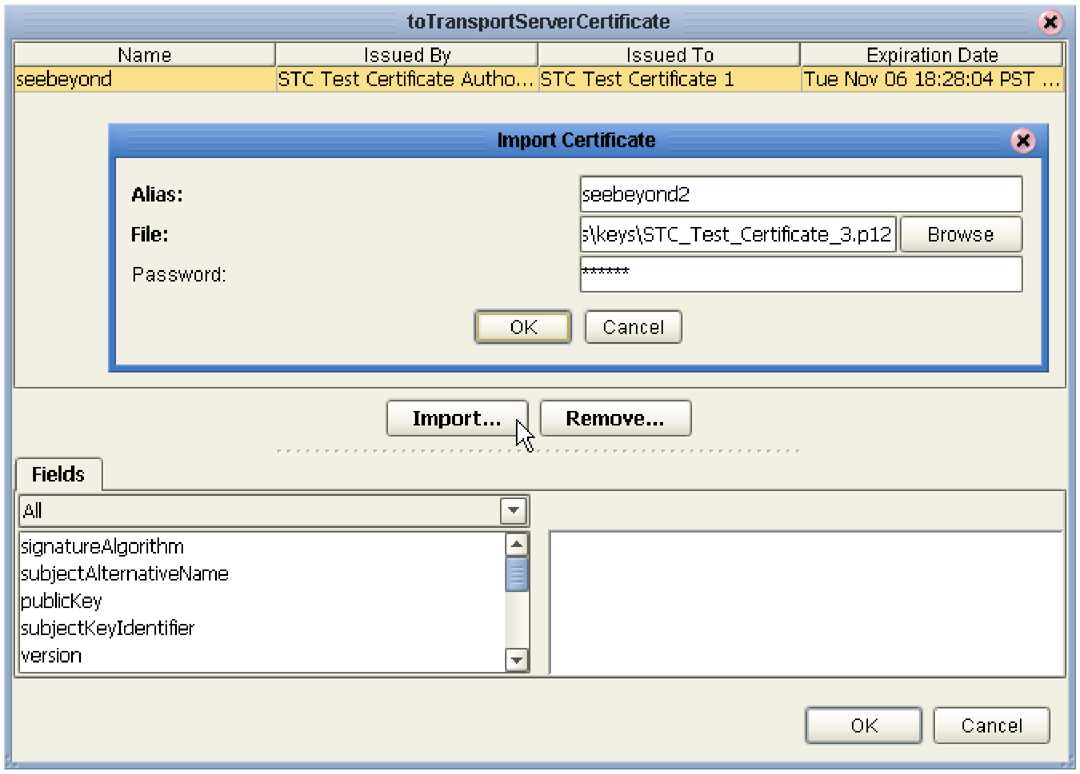
Before you begin: You must have already installed Secure Messaging Extension (see step 9 in the **procedure on page 28**), and you must have access to the locations and names of the necessary certificates and keystores.

1 In Enterprise Designer with the **Environment Explorer** tab active, in the tree, right-click the eXchange Service and, on the popup context menu, click **Properties**.

The properties page opens, showing all external delivery channels for this host.

2 One by one, as needed, click the ellipsis [**...**] button for the type of certificate or truststore you want to define for a particular external delivery channel, and either select from the drop-down list or else click **Import** to import a new certificate or truststore. For certificate import, see Figure 48; for truststore import, see Figure 49.

Figure 48 Importing a Certificate for a Delivery Channel in an eXchange Service

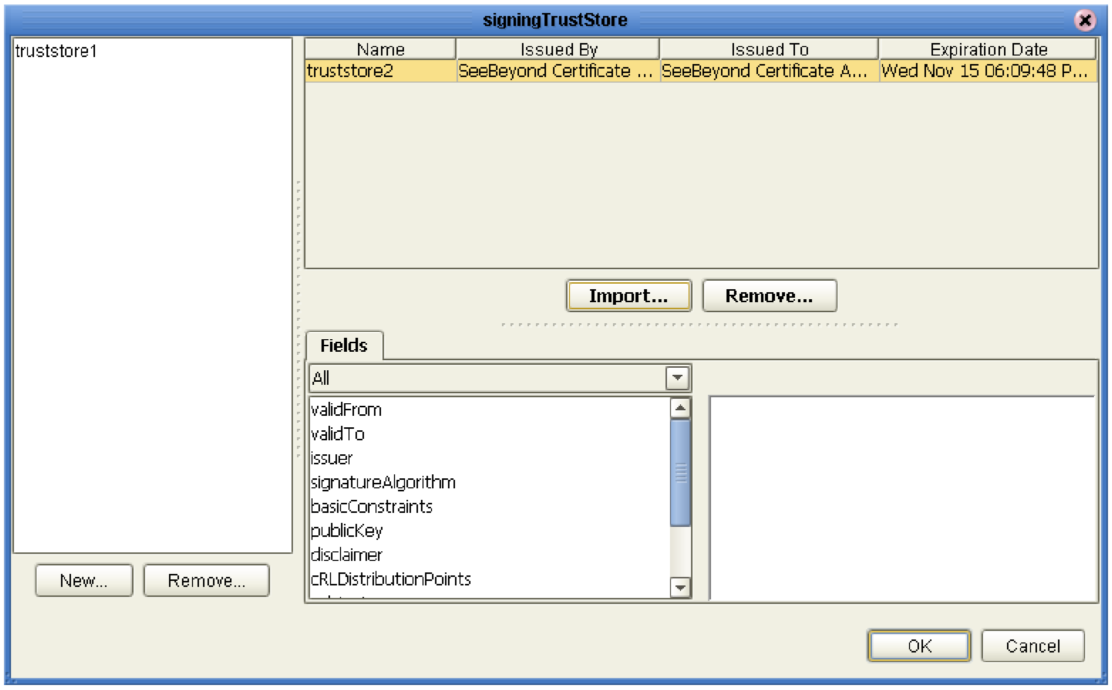

toTransportServerCertificate

Name	Issued By	Issued To	Expiration Date
seebeyond	STC Test Certificate Autho...	STC Test Certificate 1	Tue Nov 06 18:28:04 PST ...

Import Certificate

Alias: seebeyond2
File: s:\keys\STC_Test_Certificate_3.p12
Password: *******

Figure 49 Importing a TrustStore for a Delivery Channel in an eXchange Service



Name	Issued By	Issued To	Expiration Date
truststore2	SeeBeyond Certificate ...	SeeBeyond Certificate A...	Wed Nov 15 06:09:48 P...

3 Click OK to close the dialog box.

4 If a logical host were running, you would also need to do the following any time you add or update crypto information associated with the eXchange service:
In the environment tree, right-click the logical host and, on the menu, click **Apply**

Result: Cryptographic information is associated with this delivery channel. If your B2B host (and thus its eXchange Service) were associated with multiple delivery channels, each one could be configured with its own crypto information.

5.6.4 Configuring an Environment to Use HTTPS

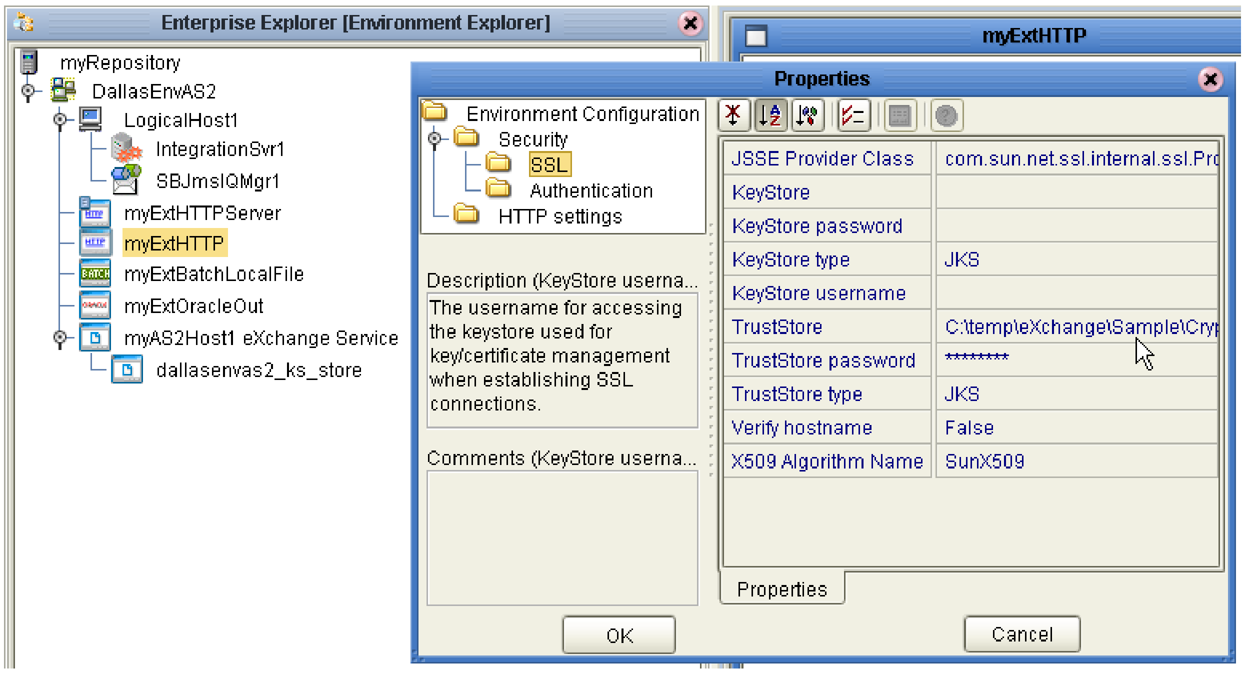
HTTPS means "HTTP over SSL" (secure sockets layer). If you want to use the HTTPS transport attributes definition, you must take additional steps to enable HTTPS.

- To enable an HTTP external to use SSL, it must have its SSL configuration settings edited appropriately. See the *HTTP(S) eWay Intelligent Adapter User's Guide*, especially the "Setting HTTP(S) eWay Properties" chapter. For an example of settings specific to eXchange, see the **procedure on page 96**.
- To enable the integration server to communicate using HTTPS, it needs to be associated with appropriate configuration settings. See the *eGate Integrator System Administration Guide* chapter on ICAN Security Features, especially the "Using SSL/ HTTPS in ICAN" section. For an example of settings specific to eXchange, see the **procedure on page 97**.

To configure the HTTP external to use SSL

1. In the Environment Explorer tree, right-click the HTTP external and, on the popup context menu, click **Properties**.
2. In the Properties sheet, open Security and click **SSL**.
3. For TrustStore, provide the path and filename of the default truststore to be used when establishing SSL connections. For example: **C:\temp\eXchange\Sample\AS2\Crypto\companyb.ssl.keystore**
4. For TrustStore password, provide the correct password for this truststore. For example: **companyb** (see Figure 50). For TrustStoreType, keep the default: **JKS**

Figure 50 Configuring the SSL Properties of the HTTP External



Description (KeyStore userna...
The username for accessing the keystore used for key/certificate management when establishing SSL connections.

Comments (KeyStore userna...

JSSE Provider Class	com.sun.net.ssl.internal.ssl.Pro
KeyStore	
KeyStore password	
KeyStore type	JKS
KeyStore username	
TrustStore	C:\temp\eXchange\Sample\Cry
TrustStore password	********
TrustStore type	JKS
Verify hostname	False
X509 Algorithm Name	SunX509

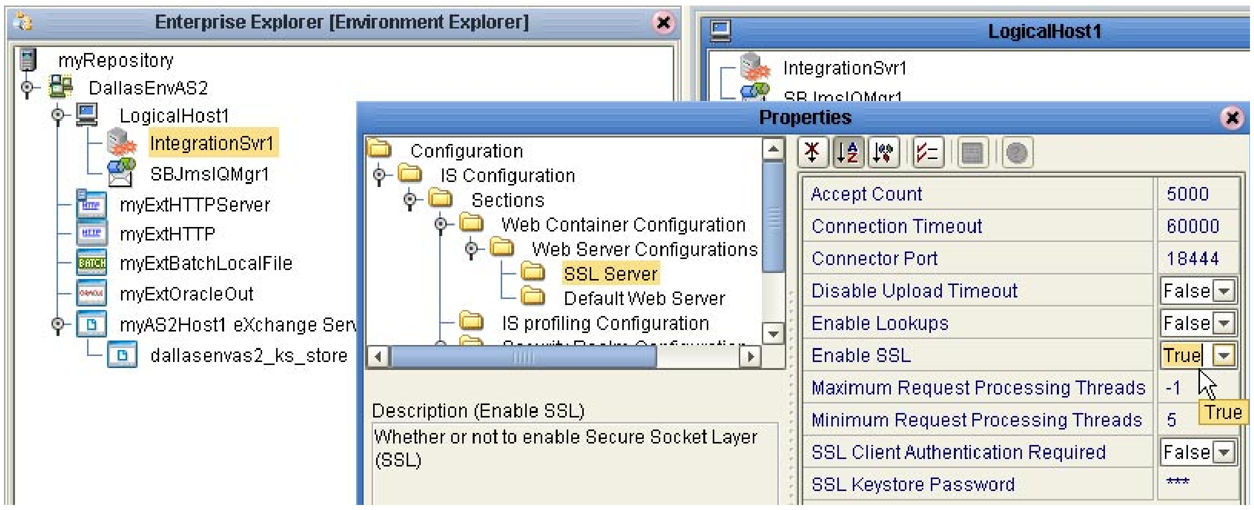
5. *AIX only.* Make the following additional changes for logical hosts running on AIX.
   - Change JSSE Provider Class from com.sun.net.ssl.internal.ssl.Provider to: **com.ibm.jsse.IBMJSSEProvider**
   - Change X509 Algorithm Name from SunX509 to (case-sensitive): **IbmX509**
6. Also make other configuration changes as needed. For more information, consult the *HTTP(S) eWay Intelligent Adapter User's Guide*, especially the "Setting HTTP(S) eWay Properties" chapter.
7. When you are finished, click OK.

Important: *Before you bootstrap the logical host, ensure that its ...\keystore\ directory contains the correct* ***.keystore*** *file and that its alias (unless you edit server.xml otherwise) is* ***tomcat****. For an example, copy <ican50>\repository\server\sbyn.keystore *to your <ican50-logicalhost>\keystore\ directory and rename it from sbyn.keystore to <yourIntegrationServername>****.keystore****. The keystore file must be of type JKS.*

To configure the integration server to use SSL

1. In the Environment Explorer tree, right-click **IntegrationSvr1** and, on the popup context menu, click **Properties**
2. In the Properties sheet, open Configuration > IS Configuration > Sections > Web Container Configuration > **Web Server Configurations**
3. Do one of the following:
   - *(not recommended)* To have your integration server always use SSL web service connections by default, change the setting for **Enable SSL** to **True** and, if needed, change the setting for **SSL Client Authentication Required** to **False**.
   - *(recommended; see Figure 51)* Right-click Web Server Configuration and, on the popup context menu, click **Create New Section**. Name the new section SSL Server. In its properties, change the setting for **Enable SSL** to **True** (see Figure 51) and change the setting for **SSL Client Authentication Required** to **False**. Ensure that its Connector Port setting does not conflict with the default IS port settings (18000-18009) or any other ports.

Figure 51 Configuring the SSL Properties of the Integration Server



Description (Enable SSL)
Whether or not to enable Secure Socket Layer (SSL)

Accept Count	5000
Connection Timeout	60000
Connector Port	18444
Disable Upload Timeout	False
Enable Lookups	False
Enable SSL	True
Maximum Request Processing Threads	-1
Minimum Request Processing Threads	5
SSL Client Authentication Required	False
SSL Keystore Password	***

4. Also make other configuration changes as needed. For more information, consult the *eGate Integrator User's Guide* (especially the "Environments" chapter, "Integration Servers" section) and the *eGate Integrator System Administration Guide* chapter on ICAN Security Features (especially the "Using SSL/HTTPS in ICAN" section).
5. When you are finished, click **OK**.

Chapter 6

Designing B2B Protocols

You can use eXchange to configure the components depicted by each activity in your B2B protocols. This chapter provides the background information you need to create and understand eXchange B2B protocols.

6.1 Overview

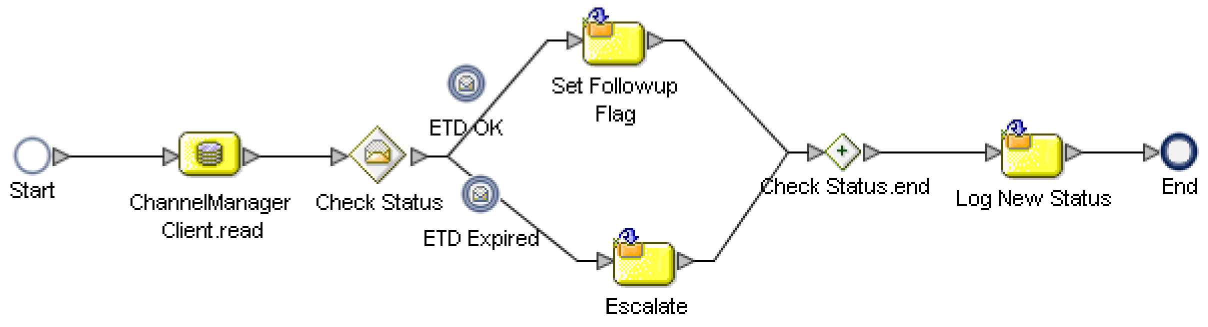
Topics in this chapter are:

6.2 Building a B2B Protocol

A business process is a collection of actions that take place in your company, revolving around a specific business practice. These processes can involve a variety of participants and may include internal and external computer systems or employees. In eXchange, you create a graphical representation of the business process called a *B2B protocol*.

A business process modeled in eXchange may look something like Figure 52.

Figure 52 Sample B2B Protocol



Add a B2B protocol to your Project

Adding a B2B protocol to your project provides an empty modeling canvas where you add and manipulate items on the canvas, called *activities*. Before you can model your business process, you must add a new B2B protocol to your project.

Note: *The eGate User's Guide has detailed information on creating a project.*

1. In Enterprise Designer, in Project Explorer, right-click the project and, on the popup context menu, point at **New** and click **B2B Protocol**.
2. Enter a new name for your B2B protocolB2B protocol.

6.2.1. Modeling a Business Process in eXchange

To model a business process in eXchange, drag and drop modeling elements on the eXchange Protocol Designer, and then link these components to reflect the logical flow of the business process. eXchange provides the tools you need to quickly develop B2B protocol models, including graphical editing tools to help you adjust, size, and align model components.

eXchange Protocol Designer

Once you create a new B2B protocol, you will build your model in the eXchange Protocol Designer (as shown in Figure 53). The eXchange Protocol Designer is the area in the Enterprise Designer where you view, create, and edit your B2B protocol.

To create a B2B protocol

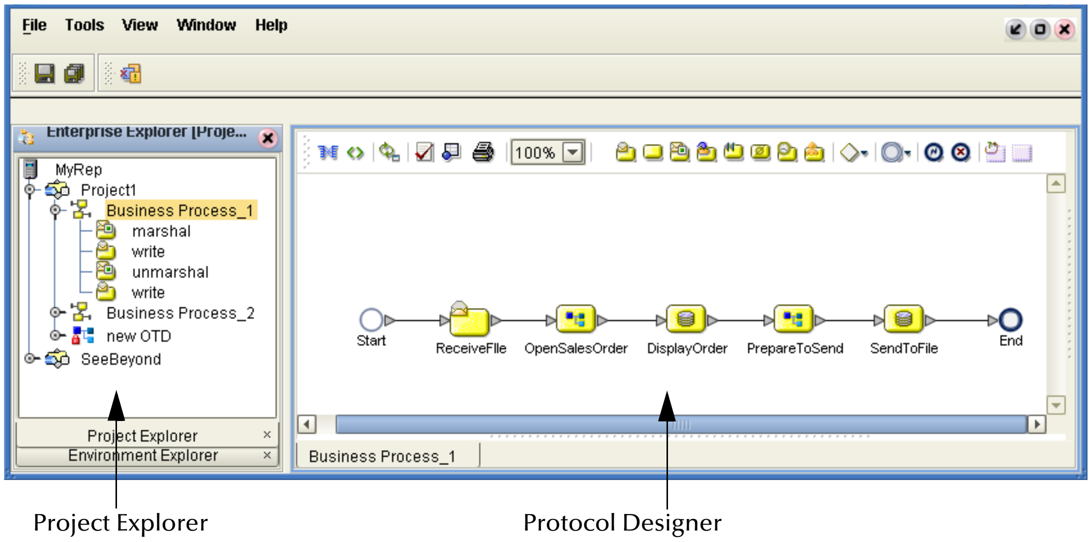
Begin designing your B2B protocol by dragging and dropping modeling elements from the toolbar onto the eXchange Protocol Designer canvas.

The **Start** and **End** activity appear on the blank canvas by default. There is only one starting point for any B2B protocol. (There can be multiple end points.)

1. Drag the appropriate modeling elements to your blank B2B protocol to the eXchange Protocol Designer canvas. See Figure 53.

Note: *See* **Appendix A** *for a complete list of modeling element options.*

Figure 53 B2B Protocol



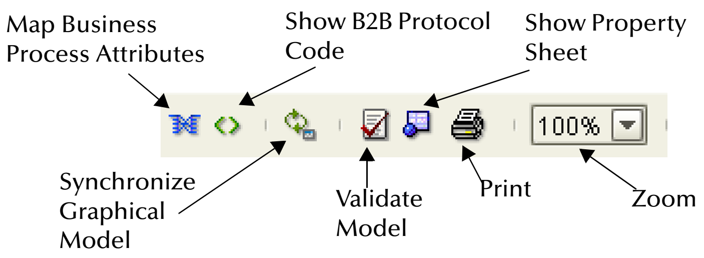
2 Draw links between the modeling elements to show the process flow (Figure 53)

3 On the main toolbar, click **Save** to save your changes to the Repository.

This will validate your B2B protocol, generate the code to run it, and save your changes to the SeeBeyond Repository.

6.3 Using the eXchange Protocol Designer GUI

Figure 54 Toolbar Options



- **Map Business Process Attributes**—Opens the **Business Rule Designer**.
- **Show B2B Protocol Code**—Toggles display of underlying BPEL code.
- **Synchronize Graphical Model and B2B Protocol Code**—Causes the graphic model, the Business Rules, and the underlying BPEL code to match.
- **Validate B2B Protocol Model**—Runs application to check syntactic validity.
- **Show Property Sheet**—Toggles display of property list and graphical overview.
- **Print**—Prints the B2B protocol graphic. Options allow you to control the scale.
- **Zoom**—Enlarges or shrinks the displayed graphic in the canvas.

6.4 Modeling Elements

The eXchange Protocol Designer is where the user creates the B2B protocol flow. It provides a palette of modeling elements for designing your B2B protocol. Like other logical components in a project, B2B protocols appear in the Project Explorer tree.

Elements from the Enterprise Explorer can either be dropped onto empty canvas or onto an Activity. Many elements provide custom settings so that you can model every detail of your process. Each B2B protocol you create consists of basic elements as described in the following sections:

- **Activity Elements** on page 101
- **Branching Activities** on page 103
- **Intermediate Events** on page 104
- **Scope** on page 105
- **While** on page 105

Activity Elements

You can include several different kinds of activities and subprocesses in a B2B protocol. For examples of each of the different kinds of activities, see Table 34.

To add an activity

1 Either drag a modeling element from the toolbar or drag a web service operation from the Project Explorer, and then drop it where you want it on the canvas.

2 Click the activity name and begin typing to rename it from the default.

Note: *Every activity name must contain at least one character (A-Z, a-z, or 0-9); it must start with a letter or an underscore (_), and it may contain spaces.*

The activity appears on the modeling canvas.

Link modeling elements

eXchange supports orthogonal and diagonal link styles – this setting applies to all links in a model and is an automated application of the style.

To link modeling elements, do the following:

1 Move your cursor over the connector portion of your modeling element.

2 Hold the cursor over the outside edge of the modeling element until it changes from the arrow pointer to a hand (see Figure 55).

Figure 55 Selected Activity

3 Click down, and drag a line from the first activity to the connector of the second activity. When the link attaches, release the mouse button.

Table 34 Activity Elements

Button	Command	Function
Start	Start Node	The Start Node is a modeling element indicating the start of the process. This element appears in the eXchange Protocol Designer by default, when you create a new B2B protocol. A Start Node can only link to an activity that has a **receive** or **read** capability, signaled by a subicon in the upper left resembling an opened envelope (see **Receive Activity** just below).
	Link Link with Business Rule	Links indicate the flow of the B2B protocol by connecting elements together. When you select a link, a context menu allows you to configure how data is going to be passed to and from the underlying component or web service operation using B2B protocol attributes. eXchange ensures the model is being properly linked because it does not allow invalid links to connect. Links can also accept mapped values. A link with mapped values is highlighted in blue.
End	End Node	The End modeling element indicates the completed state of a B2B protocol. This element appears in the B2B Protocol Designer by default, when you create a new B2B protocol.
	Receive Activity	The Receive activity indicates the invocation of a B2B protocol or a wait state pending the arrival of an inbound message. The Receive activity represents the actual method by which a B2B protocol is initiated. For example: ▪ An eWay polls a file system or database and retrieves data that is passed to the engine, along with the indication that a B2B protocol instance has started. ▪ A user types a URL into a browser and a servlet initiates a B2B protocol by sending a message to eGate or eInsight.
	Activity	An activity is a step in the B2B protocol in which the engine will invoke a web service operation or an eGate component. Depending upon the configuration of the component, a response may or may not be required. One example would be a synchronous extraction process from a database to return the credit status of a trading partner.

Table 34 Activity Elements

Button	Command	Function
	Reply Activity	The Reply activity allows a B2B protocol to respond to the external system or user that originally invoked the B2B protocol. The original receive at the beginning of the B2B protocol is paired with the Reply at the end of the process. In cases where a message must be sent back to the caller of the B2B protocol, the Reply uses information that correlates the message in the calling system. A Reply acts as the last step in a B2B protocol in which the B2B protocol is acting as a web service operation or subprocess. A Reply correlates the outbound message back to the calling process; for example, it can reply to an external system as a web service operation.
	Business Rule Activity	The Business Rule activity sets data values, including task assignments. It is used when imported models have multiple data mappings between the invocation of human tasks or automated systems.
	Compensate	The Compensate element invokes compensation on an inner scope that has already completed normally. This construct can be invoked only from within a fault handler or another compensation handler.
	Empty Activity	The Empty activity allows data to pass through without any changes.
	Wait Activity	The Wait activity acts as a timer. The user will build a model in which there are two simultaneous paths within a set scope, one for the B2B protocol and one for the timer. If the timer condition takes place first, an exception will be thrown and handled, and the B2B protocol path will then be abandoned.
	User Activity	The User activity is used only by eInsight, and should not be placed on a canvas unless your site is licensed for eInsight was well as eXchange. It is used when assigning, escalating, or otherwise using human intervention to complete eInsight business process tasks.

Branching Activities

Branching activities are objects you add to your B2B protocols to specify the logical flow of information. eXchange provides three different kinds of branching activities: Decisions, Event Based Decisions, and Flow.

Add a Branching activity

To add a Branching activity to the modeling canvas:

1. On the toolbar, click the **Branching Activities** drop-down icon, and then release the mouse button.
2. Point at the type of Branching activity you want to add, click, and then drag the activity from the toolbar to the eXchange Protocol Designer canvas.

   The selected Branching activity appears on the modeling canvas.

Table 35 Branching Activities

◇	Decision	A Decision allows one of several possible paths to execute, based on expression logic. This element is used to create complex expressions that determine the path of the B2B protocol. It also contains the expression and connection names. Decisions allow you to define expressions that are evaluated to determine the proper B2B protocol flow. Expressions are built using the mapping interface and B2B protocol attributes.
◈	Event Based Decision	Multiple possible messages can be juxtaposed against a timeout condition to allow the type of message received to determine the appropriate B2B protocol path.
◈	Flow	Allows you to specify one or more activities to be performed concurrently.

Intermediate Events

Intermediate events are those activities that can interrupt the flow of a B2B protocol. Some intermediate events handle exceptions that may occur during your B2B protocol or compensate for exceptions that occur.

Add an Intermediate event

To add an **Intermediate event** to the modeling canvas:

1. On the toolbar, click the **Intermediate Events** drop-down icon, and then release the mouse button.
2. Point at the type of Intermediate event you want to add, click, and then drag the activity from the toolbar to the eXchange Protocol Designer canvas.

   The selected Intermediate event appears on the modeling canvas.

Table 36 Intermediate Events

⏪	Compensation Handler	Used when something in a B2B protocol fails and requires a rollback or upstream activities (like money has to be returned to the customer). On an automatic basis in the B2B protocol, upstream steps in the B2B protocol are notified that the failure has occurred and certain transactions need to be reversed, sometimes in a sequential order. The compensation handler allows you to design the process and circumstances in which the compensation takes place.
Ⓝ	Catch Named Exception	Each automated system (back-end system) or web service operation can publish their possible error codes (for instance, fault 15 is “bad data”). Those codes can be mapped to exception handlers. Each exception handler is connected to the scope that surrounds one or more steps in a B2B protocol. The components within that scope will throw the exceptions when things go wrong and the exception handler will automatically initiate the appropriate process to handle the problem.
Ⓝ	Catch All Exceptions	This exception handler is configured to handle all exceptions that occur in a scope.

Table 36 Intermediate Events

◎	Message Event	This is similar to a Receive Activity, but it occurs only in the middle of a B2B protocol. Each of these elements can be a different message.
◎	Timer Event	A timeout condition is set upon Activities, sets of Activities, or a B2B protocol as a whole, to ensure that processes complete within given amount of time. Timeout conditions also allow you to design the B2B protocol branch to take after a timeout condition takes place.

Scope

The behavior for one or more activities can be defined by a scope. A scope can provide exception handlers, event handlers, a compensation handler, and data variables. The exception handlers for the scope can be used to catch the faults caused by the possible exception responses.

▨	Scope	The Scope element allows you to apply exception handlers, compensation, and transactionality to a set of sequential or simultaneous steps in a B2B protocol.

While

▨	While	This allows you to create a looping process within a B2B protocol (for instance, a negotiation process may take several weeks, but the manager wants to review the daily status). The loop continues until the negotiation is complete, and then the B2B protocol continues.

6.4.1. Validating a B2B Protocol

After creating a B2B protocol, you can check to see if there are any problems such as activities that are not connected or an incorrect number of output links from an activity.

To check the B2B protocol for errors

- On the toolbar, click **Validate B2B Protocol Model**.

  If an error is encountered, a message box displays information about the error. If there are no errors, a message appears stating that there were no errors.

Note: *If an error message displays, see **"Saving an Unfinished B2B Protocol"** for information on repairing errors. Repairing the error may entail such items as adding logic to Decisions or adding attributes to activities.*

6.4.2. Saving an Unfinished B2B Protocol

Even if a B2B protocol is not complete and/or contains errors, you can save it as a work in progress and return to it later by doing any of the following:

- On the **File** menu, choose **Save**
- On the main toolbar, click **Save**
- On the keyboard, press **Ctrl+S**

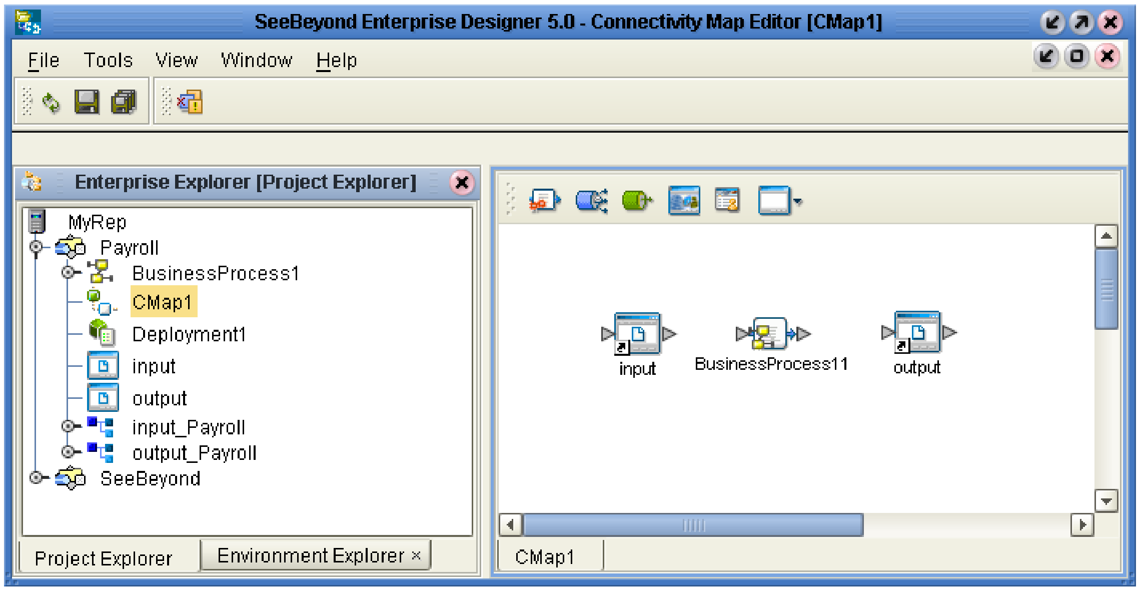
6.5 Using B2B Protocols in a Connectivity Map

The connectivity map represents connection information in the ICAN Suite. The flow is represented at a higher level than in the B2B protocol. eXchange also uses the information in the connectivity map to establish and maintain connections to systems for the correct step in a B2B protocol.

To include a B2B protocol as a service on a connectivity map

1. In the Connectivity Map Editor, drag a B2B protocol onto the canvas.
2. Add and connect other components and external systems as needed. See Figure 56.

Figure 56 Connectivity Map with B2B Protocol

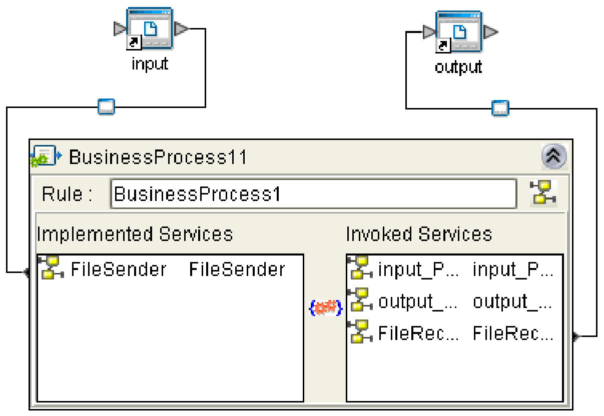


To connect the B2B protocol activities to the externals

1. In the map, double-click the B2B protocol to open the Binding Dialog.
2. Connect the appropriate activities to the corresponding external.

   Note that **Receive** activities appear in the left pane, and **Invoke** and **Reply** activities appear in the right pane. See Figure 57.

Figure 57 Connectivity Map: B2B Protocol Binding



6.6 Deploying a Project With a B2B Protocol

6.6.1. Deployment Profiles Containing B2B Protocols

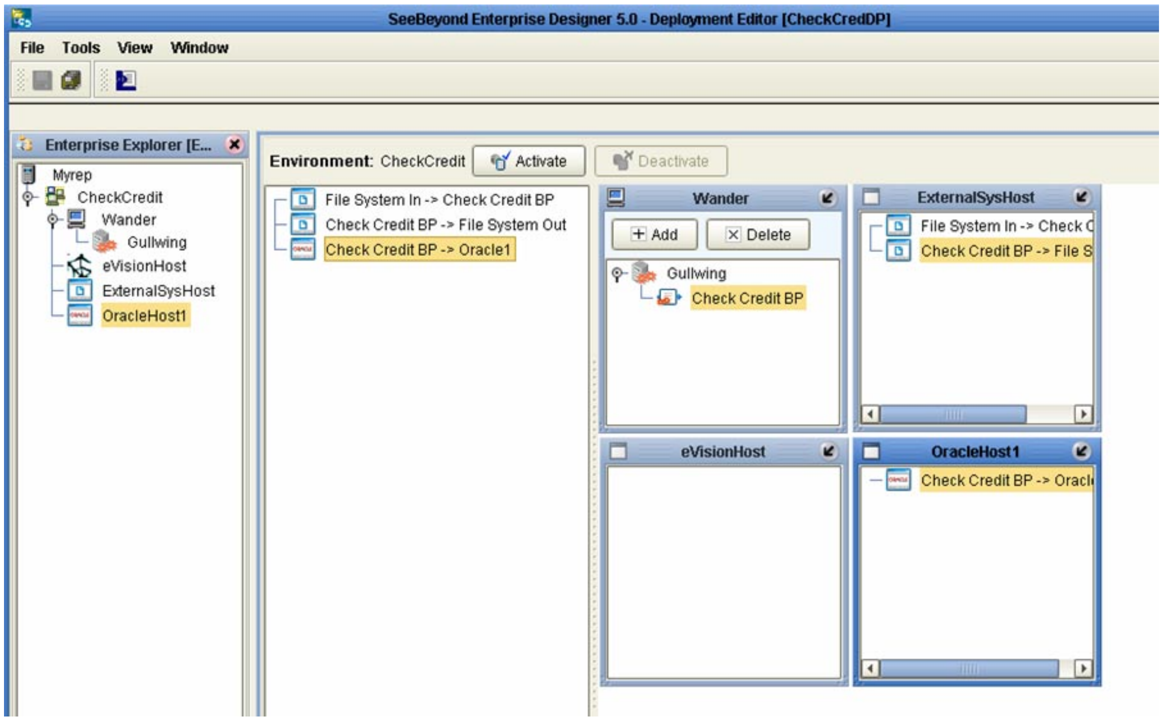
An environment becomes useful only after it has been populated with:

- One or more logical hosts, each containing one or more integration servers. Configuration steps may be required for integration servers, depending on the nature of the project and run-time environment.
- All necessary external servers, properly configured.

Note: *If a B2B host is present in a B2B protocol or connectivity map, it requires special activation steps. See* **Activating a B2B Host** *on page 90.*

When these conditions are met, a deployment profile referencing the environment assigns each logical component, eWay, and Channel Manager to a corresponding integration server and external. See Figure 58.

Figure 58 Deployment Profile with Some Components Assigned


SeeBeyond Enterprise Designer 5.0 - Deployment Editor [CheckCredDP]
File
Tools
View
Window
Enterprise Explorer [E...
Myrep
CheckCredit
Wander
Gullwing
eVisionHost
ExternalSysHost
OracleHost1
Environment: CheckCredit
Activate
Deactivate
File System In -> Check Credit BP
Check Credit BP -> File System Out
Check Credit BP -> Oracle1
Wander
Add
Delete
Gullwing
Check Credit BP
ExternalSysHost
File System In -> Check C
Check Credit BP -> File S
eVisionHost
OracleHost1
Check Credit BP -> Oracl

- Each service and B2B protocol should be dragged under the correct integration server of the appropriate logical host.
- If there are any topics or queues, each should be dragged into the correct message server of the appropriate logical host.
- Each eWay should be dragged into the correct external host system.
- Each Channel Manager, if present, should be dragged into an eXchange service for its B2B host.

After all logical components have been assigned, click **Activate** to generate the code that will be run by the integration server within the logical host engine.

Chapter 7

Exception Handling

This chapter explains the concept of exception handling and how to configure various methods of handling errors.

In this chapter

7.1 Overview

Exception handling is the identification of failed components or systems. In eXchange, exception handling allows one or more components to throw an exception that is caught by eXchange within a *scope*. Using the **scope** element, you can configure eXchange to catch all exceptions or certain exceptions that you specify. The elements that you use to configure exception handling in your model are:

- **Catch Named Exceptions**
- **Catch All Exceptions**

Exception handling in B2B protocols relies heavily on the concept of *compensation*. Compensation is an application-specific activity that reverse the effects of a previous activity that was carried out as part of a larger unit of work that is being abandoned.

B2B protocols are often of long duration and use asynchronous messages for communication. They also manipulate sensitive business data in back-end databases and line-of-business applications. As a result, the overall business transaction may fail or be cancelled after many transactions have been committed during its progress. In these cases, the partial work may need to be reversed.

7.1.1. Exception Handling Configuration

Exception handlers are configured to catch errors that are thrown by eGate components and/or Web Services. These systems can be configured to publish one or more exceptions.

- **Manual Exception Handling**: The model can contain B2B protocol logic designed to handle the exception.
- **Automatic Exception Handling**: Pre-packaged functionality guides the user to create multiple types of catches for thrown exceptions.

Each exception can be handled differently. This is one example:

1. Build the exception handling logic as a B2B protocol.
2. Select the exception handler to configure which exception triggers the exception handling process.
3. Drag the **Scope** element onto the eXchange Protocol Designer canvas.
4. Drag the **Exception** modeling element into the scope for which it should take effect.
5. Define a B2B protocol that appropriately handles each exception.
6. Model manual exceptions in a B2B protocol.
7. Configure the exception handler to take place when one of the components within the **Scope** throws the appropriate exception.

Identifying Component or System Failures

Exception management allows users to quickly identify and correct problems with components or systems.

Users can filter the list of displayed instances to quickly identify exceptions.

Users can easily navigate to particular versions of a B2B protocol to monitor the progress of instances.

A Web-based interface allows users to securely access the monitoring environment over the Internet.

Identification of troubled instances, such as time-outs or bad messages.

Failed components/systems create visual alerts via the B2B protocol monitoring interface. The integrated monitoring environment allows you to identify the problem, assign a resource to fix the problem, and if necessary, restart the affected instances.

Users can quickly identify troubled instances from a large number of instances, repair and restart that instance for continued processing.

7.2 Scope

Scope allows you to define a range

- For handling of exceptions
- For creating compensation logic

The range of the scope can span one or more activities in the B2B protocol or even the entire B2B protocol.

Scope or Process-level exceptions

Either **Catch Named Exception** or **Catch All Exceptions** can be used at the B2B protocol level.

Catch Named Exception

1. Drag the **Catch Named Exception** element into the scope for which the exception handler applies.
2. In the Exception Handler properties, configure the following:
   - Fault Container—The output Attribute that will be containing the run-time name of the thrown fault.
   - Fault Name—The run-time value for the exception that will be passed from the component to the engine at run time.

Note: *The fault name is auto-populated with values based on the components dragged to the editor.*

3. Select the configuration control for the Exception Handler – the properties pane will appear to select the Fault name and container.
4. Drag the **Catch Named Exception** into the associated scope.

Catch All Exceptions

No configuration of the Catch All Exceptions element is required; any thrown exception not previously caught is caught with the Catch All Exceptions element.

7.3 Compensation

Compensation allows the modeler to create the process flow for executing complex compensations. Exception Handlers for parent scopes invoke the correct Compensation Handlers in the appropriate order.

Using Scope and Exceptions to Trigger Compensation

- Compensation Activity—In an exception handler, initiates the compensation process. It models the compensation as a B2B protocol, and indicates the Compensation for “DB Insert” should be initiated.
- Compensation Handler— This is dropped within a scope to create the compensation logic for a given scope.

7.4 Validating the B2B Protocol

After generating the business process code (BPEL), you can click the **Validation** button on the toolbar to identify any issues with the model. The validation results now appear in a wizard, listing any issues one by one with clear and understandable descriptions for the issues. You can fix each issue, regenerate the business process code, and again view the validation results until each of the issues has been fixed, and the model validates as correct.

Chapter 8

Using eXchange Web Facilities

This chapter provides step-by-step procedures for using the Web-based facilities to configure and monitor eXchange components and processes.

In this chapter

Each of these three facilities requires that you have already created a valid project and activated it.

8.1 eXchange Partner Manager (ePM)

Before you begin: You must have already set up a B2B host with one or more delivery channels (see **"Setting Up External Delivery Channels" on page 87**) and you must have already activated it to create an eXchange Service (see **"Activating a B2B Host" on page 90**). If you want to use encryption in the trading partner, you must have already configured the proper certificates and truststores in the environment properties of the eXchange Service (see **procedure on page 94**, **"To associate security information with an eXchange Service's delivery channels"**).

Overview

In this procedure, you will add a trading partner, bind it to at least one external delivery channel, add a profile beneath it, configure the profile, assign a messaging service to it, and configure all of the actions within the messaging service.

Accessing eXchange Partner Manager (ePM)

To access eXchange Partner Manager and locate a B2B host in the explorer tree

Before you begin: Your repository server must already be running.

1. Open a browser window and point it at the hostname and port where your Repository server is running. For example:

   ```
   http://localhost:16271
   ```

2. Log in to Enterprise Manager.

3 When you have logged in, point your browser at a new URL that is the same as the previous one but with the string **/epm** appended. For example:

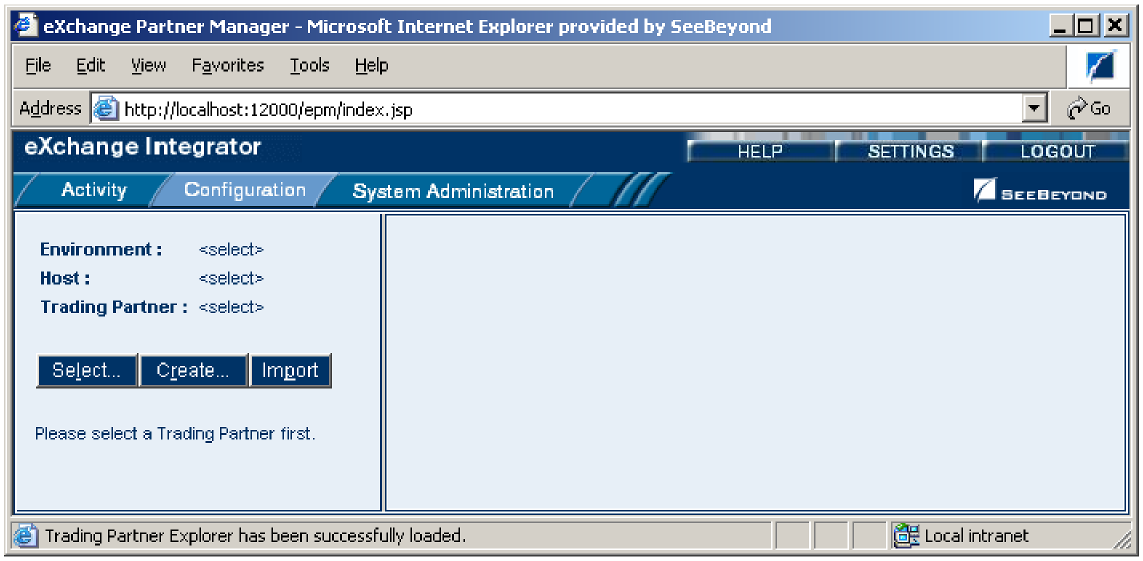
```
http://localhost:16271/epm
```

Note: *This URL is case-sensitive. If your browser warns you, "Requested resource not available," double-check that you entered the suffice **/epm** in all-lowercase.*

4 Click **Sign In**.

Result: After a pause, the window displays eXchange Partner Manager (ePM), a two-pane window with an explorer tree and a canvas. See Figure 59.

Figure 59 eXchange Partner Management (ePM) — Initial State


eXchange Partner Manager - Microsoft Internet Explorer provided by SeeBeyond
File Edit View Favorites Tools Help
Address http://localhost:12000/epm/index.jsp
Go
eXchange Integrator
HELP SETTINGS LOGOUT
Activity Configuration System Administration
SEEBEYOND
Environment : <select>
Host : <select>
Trading Partner : <select>
Select... Create... Import
Local intranet

Please select a Trading Partner first.

Trading Partner Explorer has been successfully loaded.

Creating Trading Partners

To create and name a trading partner

1 In the tree, click **Create**; then, in the **Create a New Trading Partner** window:

A Open the B2B Repository and environment and select the B2B host.

B Enter a name for the trading partner to create.

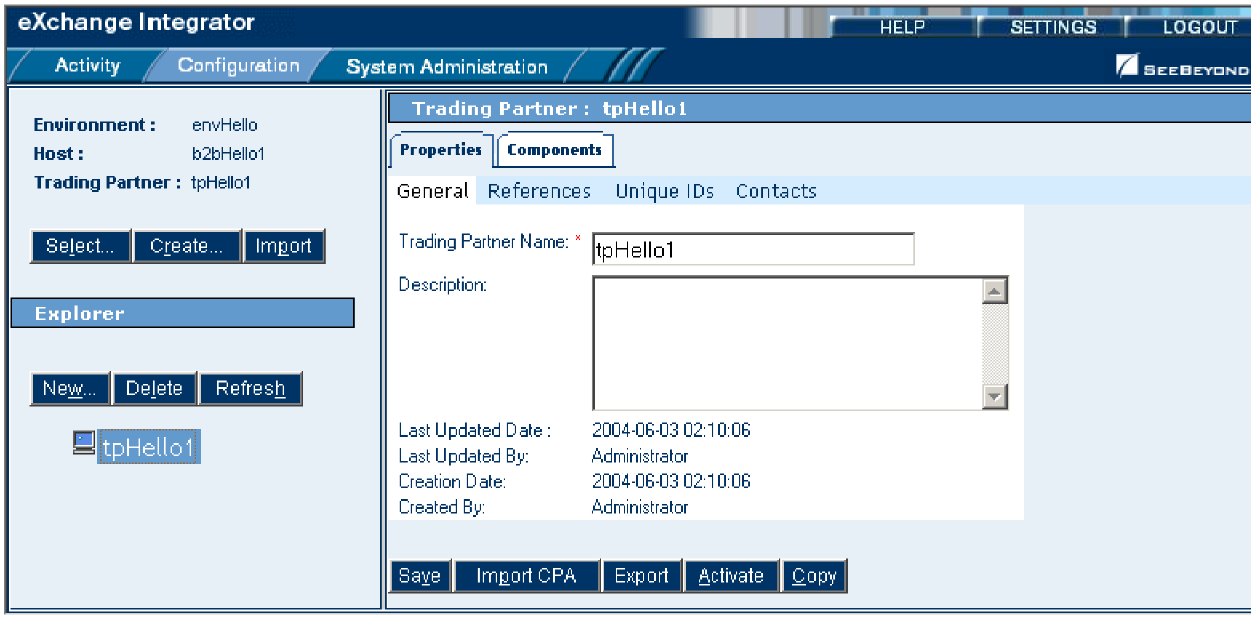
C Click **Create**.

Result: In the explorer tree, the new trading partner appears under the B2B host.

2 Click the trading partner.

The **Trading Partner Configuration** canvas opens, displaying two tabs: **Properties** and Components. See Figure 60.

Figure 60 New Trading Partner: Properties Tab



Environment : envHello
Host : b2bHello1
Trading Partner : tpHello1

Trading Partner : tpHello1

Trading Partner Name: * tpHello1
Description:
Last Updated Date : 2004-06-03 02:10:06
Last Updated By: Administrator
Creation Date: 2004-06-03 02:10:06
Created By: Administrator

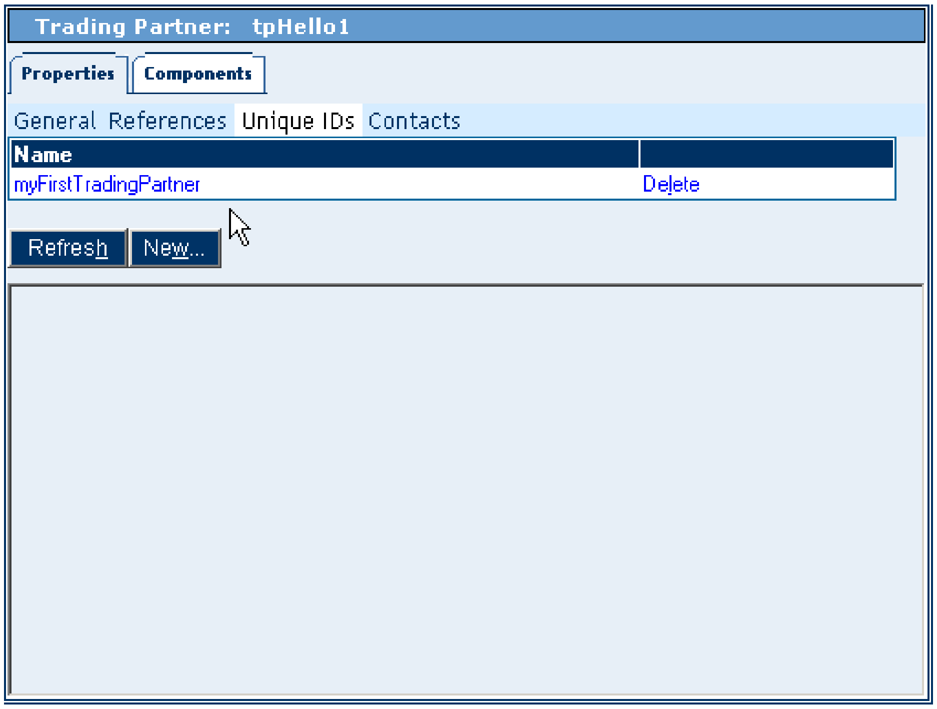
3 In the **Properties** tab with the **General** subtab active, click **Save**.

4 Click **Unique IDs** (the third subtab), and then click **New** (center-right button).

5 Enter a unique ID for this trading partner, and then click **Save**.

The new unique ID appears under the Name column. See Figure 61.

Figure 61 New Trading Partner with Unique ID



Trading Partner: tpHello1

Name	
myFirstTradingPartner	Delete

Creating Bindings to External Delivery Channels

To access an external delivery channel and create a binding to it

1 With a trading partner highlighted, in the configuration canvas, click **Components**.

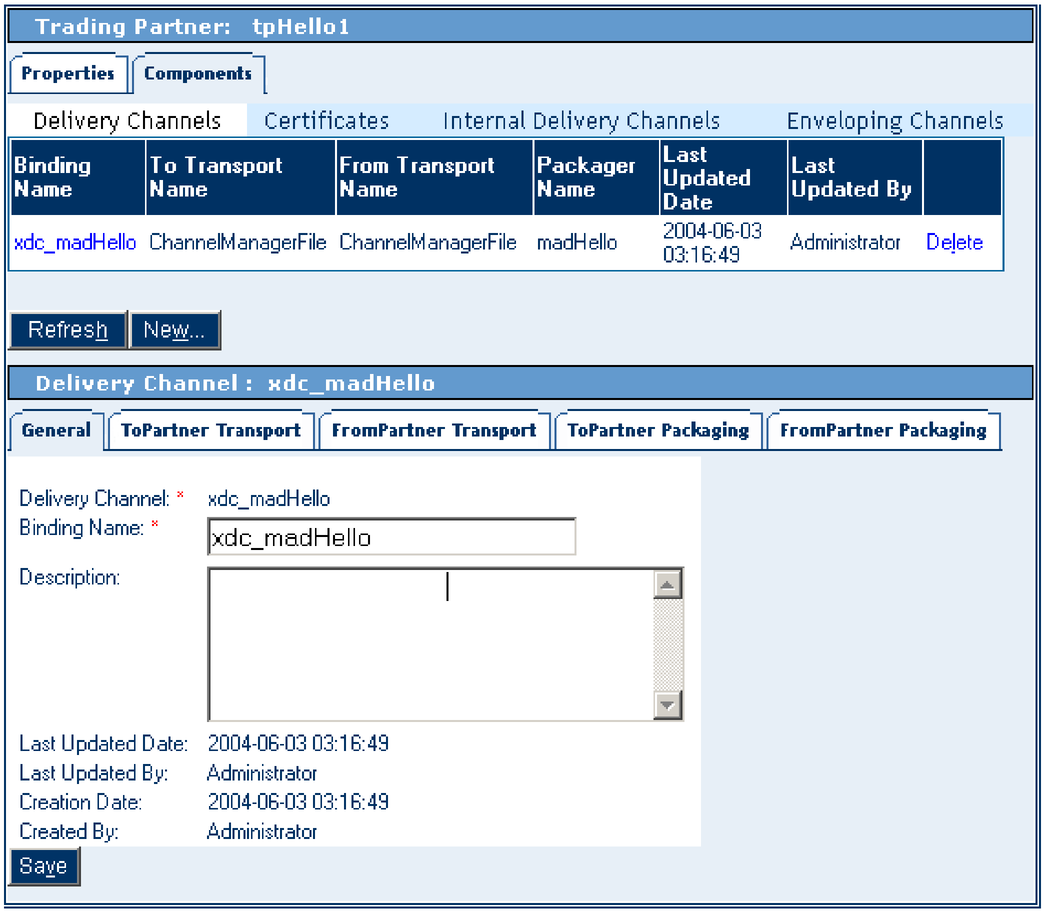
When the Components tab is active, it displays four subtabs: **Delivery Channels**, Certificates, Internal Delivery Channels, and Enveloping Channels.

2 With **Delivery Channels** active (the first subtab), click **New** (center-right button).

3 Select a delivery channel from the list, enter or accept a binding name, and click **Continue**.

When the delivery channel's five subtabs appear (**General**, **ToPartner Transport**, **FromPartner Transport**, **ToPartner Packaging**, and **FromPartner Packaging**.), click **Save**. See Figure 62.

Figure 62 Trading Partner Newly Associated with an External Delivery Channel



Binding Name	To Transport Name	From Transport Name	Packager Name	Last Updated Date	Last Updated By	
xdc_madHello	ChannelManagerFile	ChannelManagerFile	madHello	2004-06-03 03:16:49	Administrator	Delete

Delivery Channel: *	xdc_madHello
Binding Name: *	xdc_madHello
Description:	
Last Updated Date:	2004-06-03 03:16:49
Last Updated By:	Administrator
Creation Date:	2004-06-03 03:16:49
Created By:	Administrator

You will need to provide specific parameter values to be used by the transport and enveloping(=packaging) protocols.

4 Click the **ToPartner Transport** tab and enter appropriate transport attributes definition values.

The number and type of parameters depends on the particular transport attributes definition you are using. For example, a File transport attributes definition might require only three values, an FTP transport attributes definition might require six and permit twelve, and a custom transport attributes definition might require hundreds of values, or none.

5 When you have entered all values, click **Save**.

6 Repeat the previous two steps for the other three tabs (FromPartner Transport, ToPartner Packaging, and FromPartner Packaging). Be sure to click **Save** each time.

Tip: *For the ToPartner_Packaging and FromPartner_Packaging tabs, be sure to open them and click **Save** even if you do not enter any values.*

Result: The binding for this external delivery channel has been configured.

Optionally, if you are using Secure Messaging Exchange (SME), you can import an Encryption Key in the **ToPartner Packaging** tab and import a Signature certificate in the **FromPartner Packaging** tab.

To import a signature certificate or encryption key

1 Do one of the following:

- In a Components Delivery Channels > **[ ]Partner Packaging** tab, click **Import**.
- In the Components >**Certificates** tab, click **New**.

2 In the **Import a Certificate** window, enter a value for **Certificate Name**, click **Browse**, locate and select the correct certificate, and then click **OK**.

3 Click **Save**.

Result: The imported certificate is saved. It is now displayed both under the **Certificates** tab and in the drop-down list.

Creating Bindings to Internal Delivery Channels

To access an internal delivery channel and create a binding to it

1 With a trading partner highlighted, in the configuration canvas, click **Components**.

When the Components tab is active, it displays four subtabs: Delivery Channels, Certificates, **Internal Delivery Channels**, and Enveloping Channels.

2 Click **Internal Delivery Channels** and then click **New** (center-right button).

3 Select an IDC from the list, enter or accept a binding name, and click **Continue**.

Two new subtabs are displayed—in addition to **General**, one or the other of:

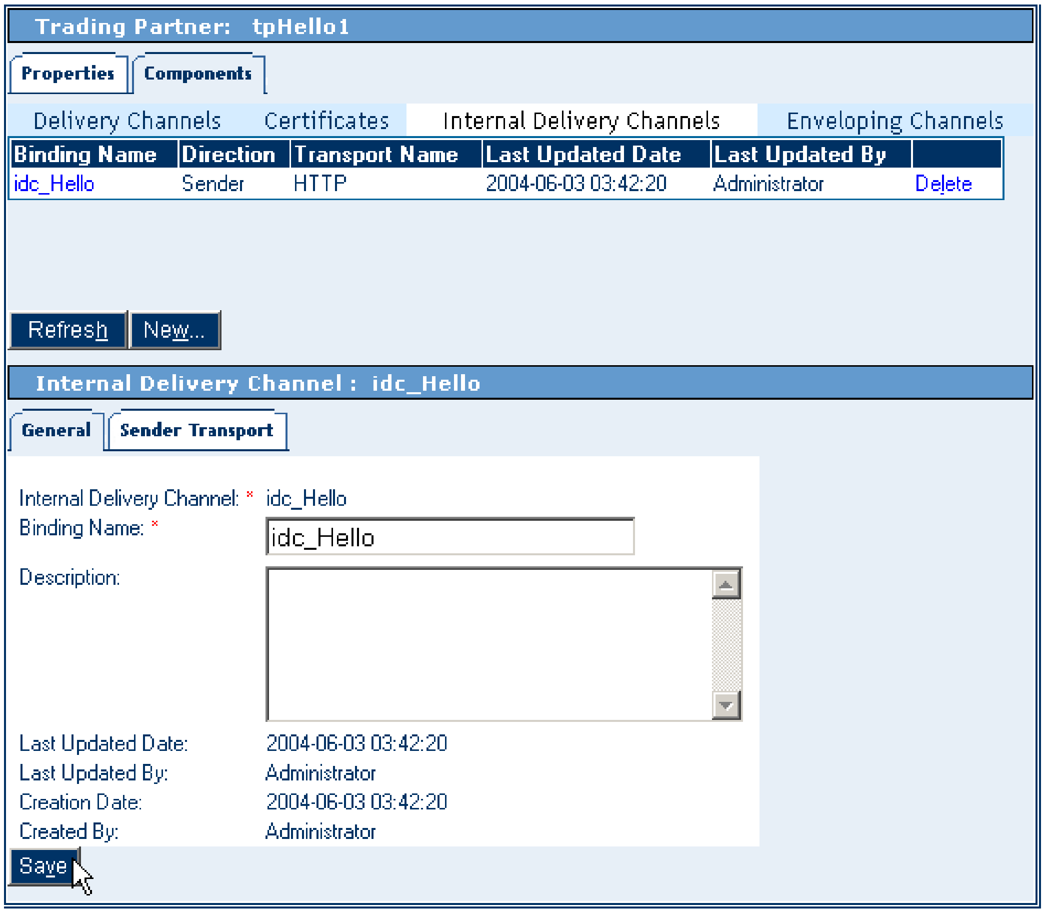
- **Sender Transport** (for IDCs that the B2B host designated as **Sender**)
- **Receiver Transport** (for IDCs that the B2B host designated as **Receiver**)

4 Click the **Sender Transport** or **Receiver Transport** tab and supply values for that transport attributes definition (if any; for example, the HTTP transport attributes definition takes no input values).

5 Click **Save**. See Figure 63.

Figure 63 Trading Partner Newly Associated with an Internal Delivery Channel

1



Binding Name	Direction	Transport Name	Last Updated Date	Last Updated By	
idc_Hello	Sender	HTTP	2004-06-03 03:42:20	Administrator	Delete

Internal Delivery Channel: *	idc_Hello
Binding Name: *	idc_Hello
Description:	
Last Updated Date:	2004-06-03 03:42:20
Last Updated By:	Administrator
Creation Date:	2004-06-03 03:42:20
Created By:	Administrator

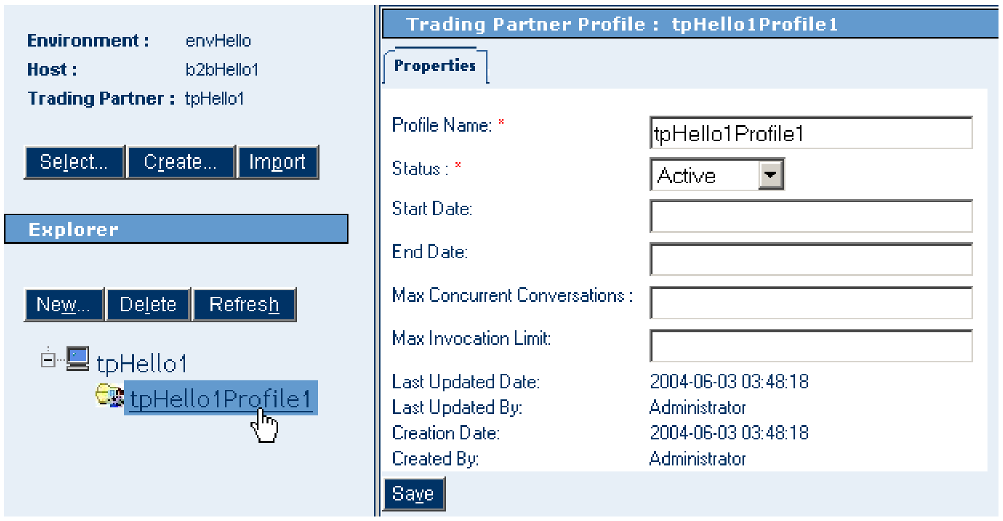
Creating Profiles and Activating Trading Partners

To create a trading partner profile and activate the trading partner

1. In the explorer tree, with a trading partner highlighted, click **New** (far left button).
2. Enter a profile name, select a status (if not **proposed**), and optionally enter a start date, end date, and maximum values concurrent conversations and invocations.
3. Click **Save**.

   The explorer tree displays a new trading partner ***profile***. See Figure 64.

Figure 64 Newly Created Trading Partner Profile



Environment :	envHello
Host :	b2bHello1
Trading Partner :	tpHello1

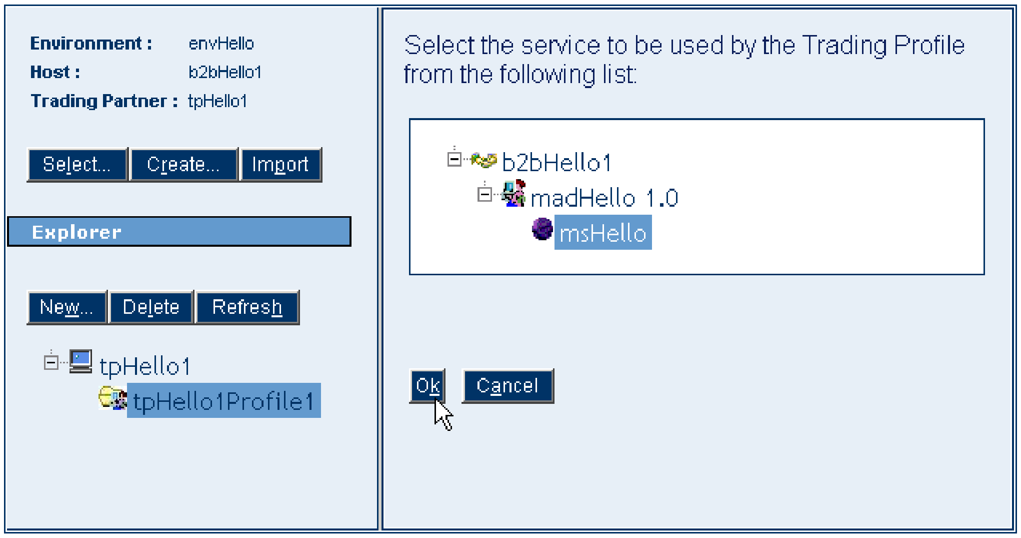
Trading Partner Profile : tpHello1Profile1

Profile Name: *	tpHello1Profile1
Status : *	Active
Start Date:	
End Date:	
Max Concurrent Conversations :	
Max Invocation Limit:	
Last Updated Date:	2004-06-03 03:48:18
Last Updated By:	Administrator
Creation Date:	2004-06-03 03:48:18
Created By:	Administrator

4 In the explorer tree, click the newly created profile, and then click **New** (far left).

You are creating a new messaging service binding based on one of the messaging services that are the leaves of the tree now displayed on the canvas. See Figure 65.

Figure 65 Messaging Services Organized Under Enveloping Attributes Definitions



Environment :	envHello
Host :	b2bHello1
Trading Partner :	tpHello1

Select the service to be used by the Trading Profile from the following list:

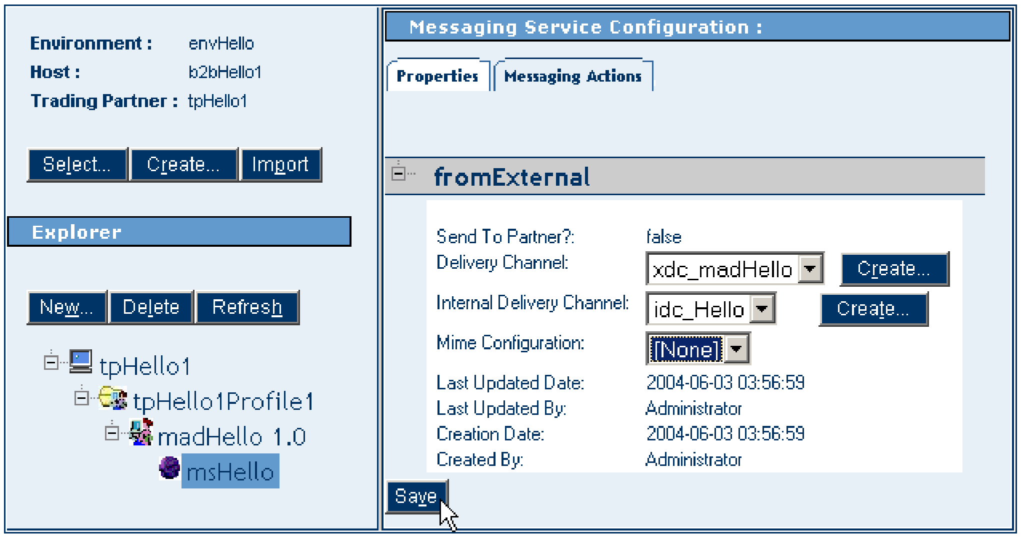
5 Click one of the messaging services and then click **OK**.

The messaging service appears as the bottommost leaf of the explorer tree, under the messaging attributes definition (standard or custom) with which it is associated.

6 Click **Save**, and then click the **Messaging Actions** tab to display the actions that you defined for that messaging service.

7 Open each action, assign appropriate values, and then click **Save**. See Figure 66.

Figure 66 Messaging Service in Explorer Tree with Actions Shown on Canvas



Environment : envHello
Host : b2bHello1
Trading Partner : tpHello1

Send To Partner?: false
Delivery Channel: xdc_madHello
Internal Delivery Channel: idc_Hello
Mime Configuration: [None]
Last Updated Date: 2004-06-03 03:56:59
Last Updated By: Administrator
Creation Date: 2004-06-03 03:56:59
Created By: Administrator

8 In the explorer tree, click the trading partner to display the Trading Partner canvas.

9 Click **Activate** (lower-right button). In response to the prompt, click **Activate**.

A success message confirms that the trading partner configuration is now in the database, and accessible to Channel Manager and other partner lookup facilities.

8.2 Monitoring B2B Protocols

You use standard eGate tools within Enterprise Manager to monitor your B2B protocols.

Before you begin

- You must already have activated a project whose connectivity map contains one or more services for B2B protocols.
- You must already have bootstrapped a logical host to run this project.

To monitor a B2B protocol

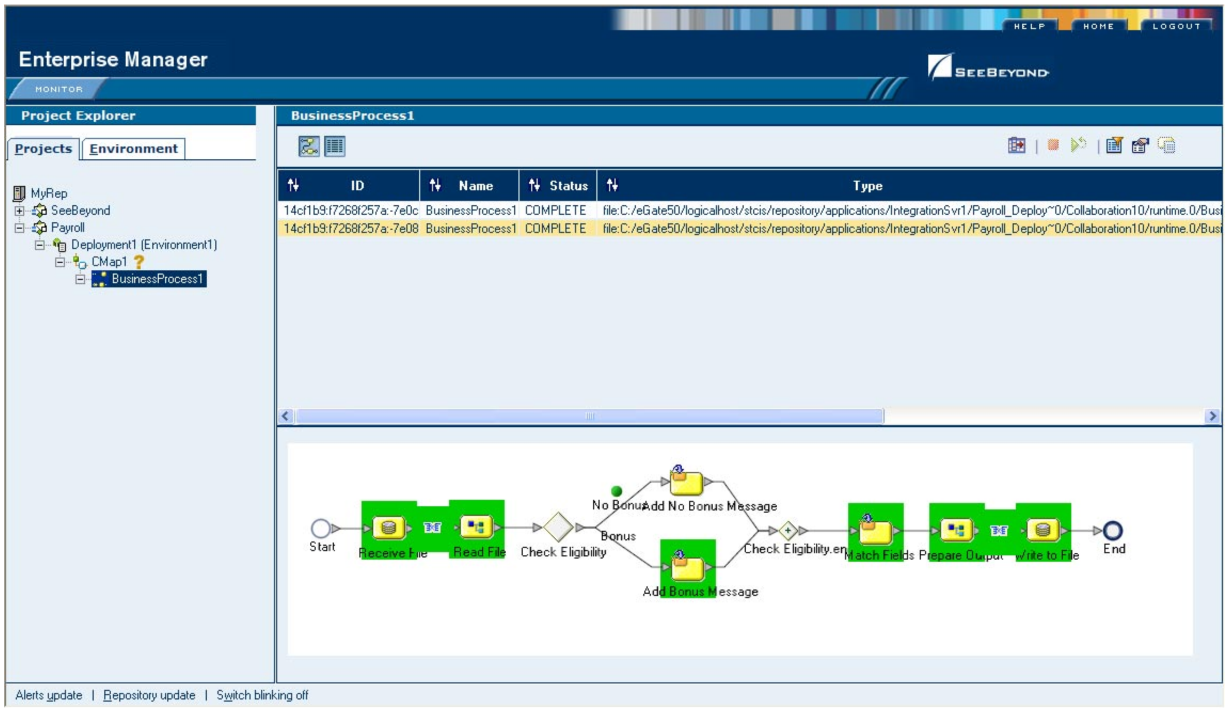
1 Start **Enterprise Manager** and click the **HOME** tab on the far left.

2 Select the **ICAN Monitor** icon to bring up the tree structure which allows you to navigate through projects or environments.

3 Select the **Project** tab and, in the tree, open the project, [subproject,] deployment profile, and connectivity map of the B2B protocol you want to monitor.

4 On the canvas (right pane), under either the **Graphic** tab or the **List** tab, select the B2B protocol.

Figure 67 Monitor View



ID	Name	Status	Type
14cf1b9:f7268f257a:-7e0c	BusinessProcess1	COMPLETE	file:C:/eGate50/logicalhost/stcis/repository/applications/IntegrationSvr1/Payroll_Deploy~0/Collaboration10/runtime.0/Busi
14cf1b9:f7268f257a:-7e08	BusinessProcess1	COMPLETE	file:C:/eGate50/logicalhost/stcis/repository/applications/IntegrationSvr1/Payroll_Deploy~0/Collaboration10/runtime.0/Busi

Monitor options

In the Monitor canvas, the four top tabs are: Alerts, Logging, Lists, and Controls:

- **Alerts**—Displays all alerts for the component selected in the explorer tree.
- **List**—Displays a list of how the components relate to one another.
- **Logging**—Displays all log messages for the selected component. Further options allow you to filter and search log messages.
- **Controls**—Displays controls that allow you to stop and start components.

For detailed information on the ICAN Monitor, see the *eGate Integrator User's Guide* and the *eGate Integrator System Administration Guide*.

8.3 Message Tracking

eXchange provides a special application, named Message Tracker, that allows you to monitor the status of messages as they are received and processed by eXchange.

Before you begin

- You must already have activated a project whose connectivity map contains one or more instances of the eXchange Tracker_Application.
- Your Oracle database for eXchange 5.0.4 must already be running, and you must already have bootstrapped a logical host to run this project.
- For the facility to be useful, there must be one or more messages that have already been picked up by this logical host's integration server.

Accessing eXchange Message Tracking

To access Message Tracking

1. Start a *new* browser session (that is, do ***not*** clone a window of an existing session).
2. Point your browser at the following URL

   http://<*loghostname*>**:**<*port*>**/**<*appname*>**/msgTrack/EnterPkgTrack.do**

   where:

   - <*loghostname*> is the hostname or IP address of a logical host running your project.
   - <*port*> is the Web server connector port configured in your integration server. To learn this, use Environment Explorer to open the logical host; right-click the integration server and select Properties; open IS Configuration > Sections > Web Container > Web Server > Default Web Server; <*port*> is the value set for Connector Port. If you have several web server configurations, check them also.

     If you have made no changes to the defaults, the value will be **18004** (for the first integration server in the first-created logical host; 19004 for the first integration server in the second-created logical host, and so forth).
   - <*appname*> is the name of your Message Tracker application as it appears on the connectivity map.

Examples:

- To access message tracking for "LH1" (IS ports 18000–18009, web port=18004):

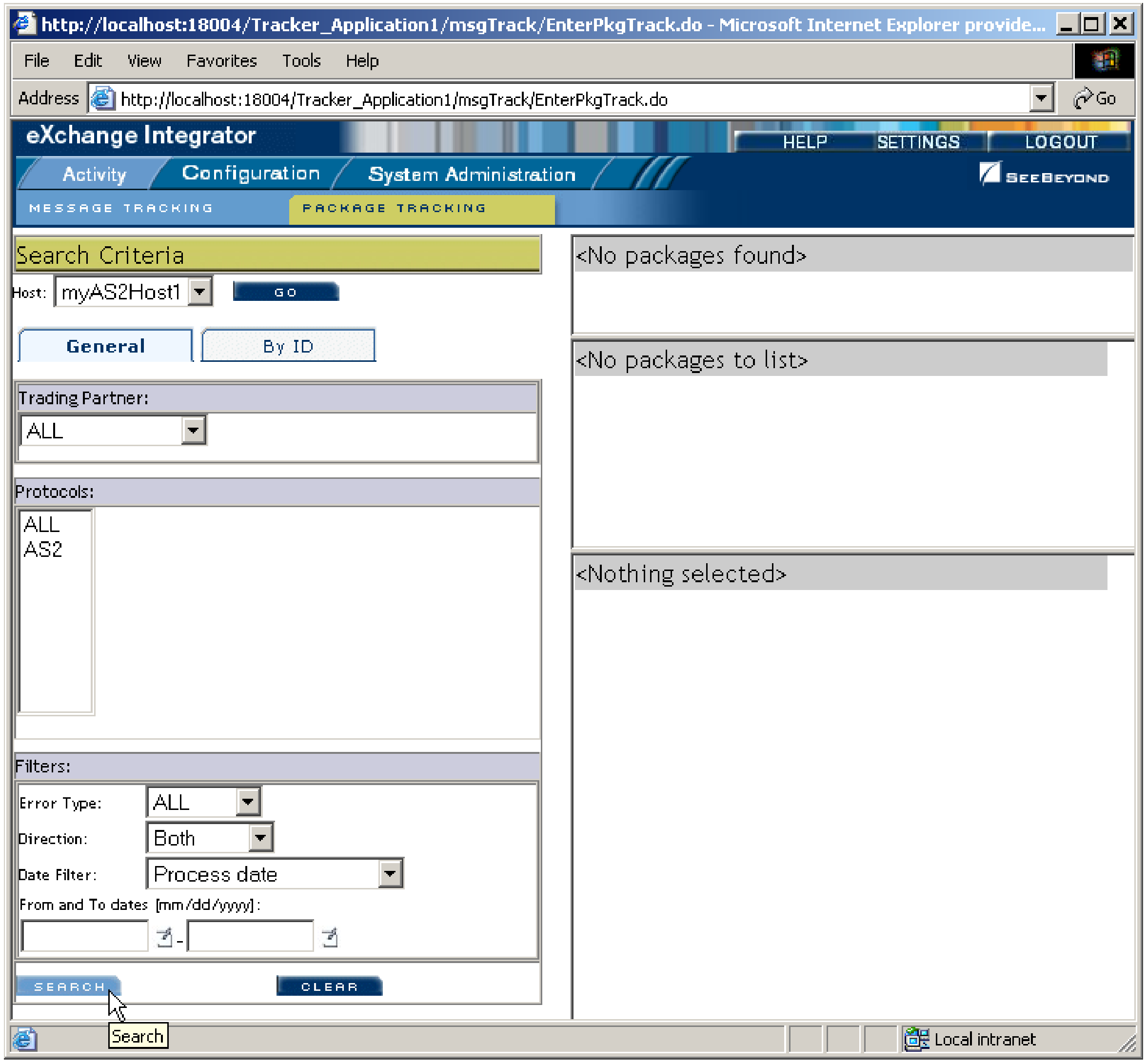
  `http://LH1:`**`18004`**`/Tracker_Application1/msgTrack/EnterPkgTrack.do`
- To access message tracking for "LH2" (IS ports 19000–19009, web port=19004):

  `http://LH2:`**`19004`**`/Tracker_Application1/msgTrack/EnterPkgTrack.do`
- Or if, instead (on LH1, web port=18004), you had named your tracking application "myTracker":

  `http://LH1:18004/myTracker/msgTrack/EnterPkgTrack.do`

Result: See Figure 68.

Figure 68 Message Tracking, on Startup


http://localhost:18004/Tracker_Application1/msgTrack/EnterPkgTrack.do - Microsoft Internet Explorer provide...
File Edit View Favorites Tools Help
Address http://localhost:18004/Tracker_Application1/msgTrack/EnterPkgTrack.do
Go
eXchange Integrator
HELP SETTINGS LOGOUT
Activity Configuration System Administration
SeeBeyond
MESSAGE TRACKING PACKAGE TRACKING
Search Criteria
Host: myAS2Host1
GO
General By ID
Trading Partner:
ALL
Protocols:
ALL
AS2
Filters:
Error Type: ALL
Direction: Both
Date Filter: Process date
From and To dates [mm/dd/yyyy]:
SEARCH
CLEAR
Search
<No packages found>
<No packages to list>
<Nothing selected>
Local intranet

8.3.1. Using Message Tracking

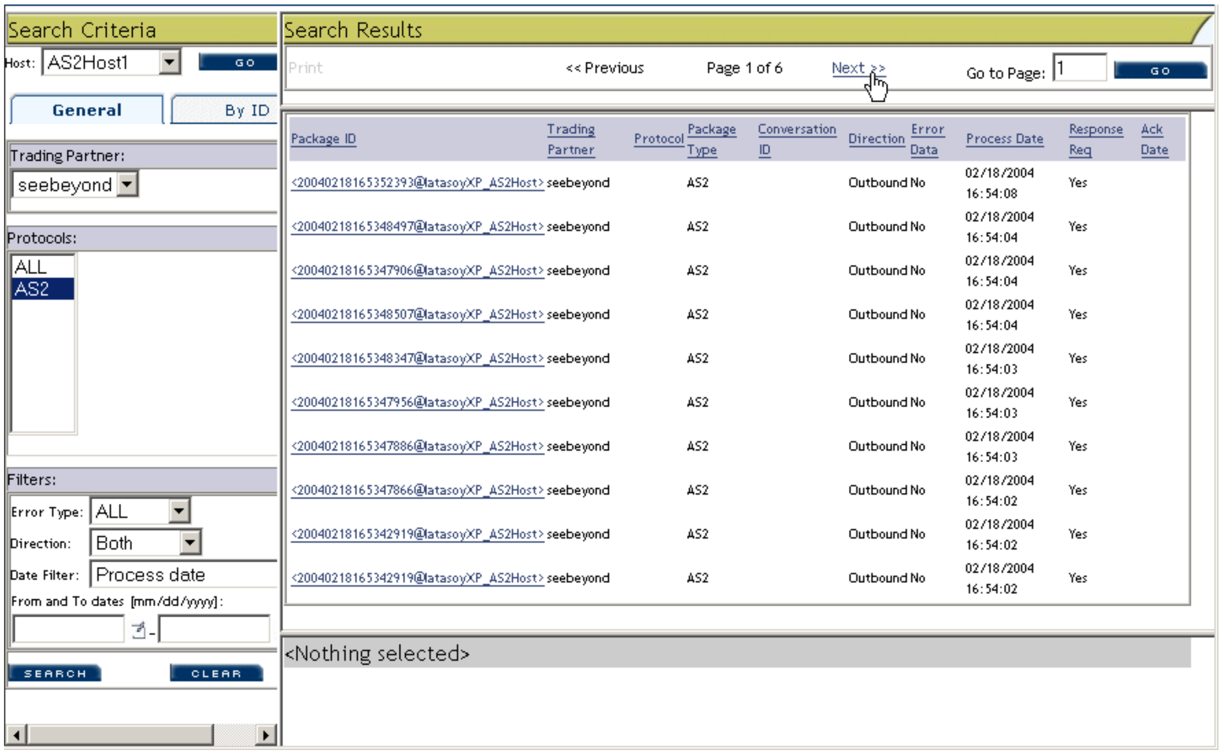
To search by B2B host, trading partner, and protocol

1. Under Search Criteria, use the Host drop-down list to choose the B2B host whose messages you want to examine, and click **GO**.
2. Under Trading Partner, either click ALL or choose a particular trading partner from the drop-down list.
3. Under Protocols, either click ALL or choose a protocol from the drop-down list.
4. At the lower left of the window, click **SEARCH**.

Result: The canvas (right side), under Search Results, displays a page containing the Package IDs of the latest ten tracked messages fitting the criteria you specified.

Navigation links (Previous, Next, and Go to Page) allow you to see other pages of ten results each. See Figure 69.

Figure 69 Message Tracking, Showing Initial Search Results



Search Criteria

Host: AS2Host1

General | By ID

Trading Partner: seebeyond

Protocols: ALL, AS2

Filters:

Error Type: ALL

Direction: Both

Date Filter: Process date

From and To dates [mm/dd/yyyy]:

Search Results

Print   << Previous   Page 1 of 6   Next >>   Go to Page: 1

Package ID	Trading Partner	Protocol	Package Type	Conversation ID	Direction	Error Data	Process Date	Response Req	Ack Date
<20040218165352393@latasoyXP_AS2Host>	seebeyond		AS2		Outbound	No	02/18/2004 16:54:08	Yes	
<20040218165348497@latasoyXP_AS2Host>	seebeyond		AS2		Outbound	No	02/18/2004 16:54:04	Yes	
<20040218165347906@latasoyXP_AS2Host>	seebeyond		AS2		Outbound	No	02/18/2004 16:54:04	Yes	
<20040218165348507@latasoyXP_AS2Host>	seebeyond		AS2		Outbound	No	02/18/2004 16:54:04	Yes	
<20040218165348347@latasoyXP_AS2Host>	seebeyond		AS2		Outbound	No	02/18/2004 16:54:03	Yes	
<20040218165347956@latasoyXP_AS2Host>	seebeyond		AS2		Outbound	No	02/18/2004 16:54:03	Yes	
<20040218165347886@latasoyXP_AS2Host>	seebeyond		AS2		Outbound	No	02/18/2004 16:54:03	Yes	
<20040218165347866@latasoyXP_AS2Host>	seebeyond		AS2		Outbound	No	02/18/2004 16:54:02	Yes	
<20040218165342919@latasoyXP_AS2Host>	seebeyond		AS2		Outbound	No	02/18/2004 16:54:02	Yes	
<20040218165342919@latasoyXP_AS2Host>	seebeyond		AS2		Outbound	No	02/18/2004 16:54:02	Yes	

<Nothing selected>

To search by B2B host, trading partner, and protocol

1. Under Search Criteria, use the Host drop-down list to choose the B2B host whose messages you want to examine, and click **GO**.
2. For Protocols, either click ALL or choose a particular protocol from the list.
3. For Package Type, either click ALL or choose a particular packaging protocol from the drop-down list.
4. For ID, enter a string for matching the message ID.
5. At the lower left of the window, click **SEARCH**.

Result: The canvas displays a page containing the Package IDs of the latest ten tracked messages fitting the criteria you specified.

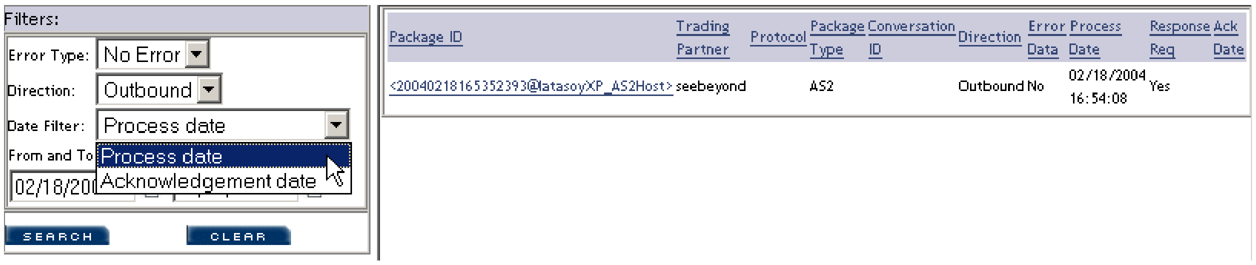
To filter results by error type, direction, and/or date

Purpose: After performing a search, or after setting up a search using either of the two previous procedures, you can specify one or more further criteria.

1. Near the bottom of the left pane, under **Filters**, specify one or more of the following:
   - For **Error Type**: If you do not choose ALL, you can restrict your search either to display error messages only, or to display non-error messages only.

- For **Direction**: If you do not choose ALL, you can restrict your search either to display inbound messages only, or to display outbound messages only.
- For **Date**: You can choose to include only those messages whose *processing* date lies within a range you specify, or only those messages whose *acknowledgment* date lies within the range. See Figure 70.

Figure 70 Message Tracking, Showing Filters



Filters:

Error Type:	No Error
Direction:	Outbound
Date Filter:	Process date
From and To	02/18/200

Package ID	Trading Partner	Protocol	Package Type	Conversation ID	Direction	Error Data	Process Date	Response Req	Ack Date
<20040218165352393@latasoyXP_AS2Host>	seebeyond		AS2		Outbound	No	02/18/2004 16:54:08	Yes	

2 At the lower left of the window, click **SEARCH**.

Result: The canvas displays a page containing the Package IDs of the latest ten tracked messages fitting the criteria you specified.

To obtain details of a specified package

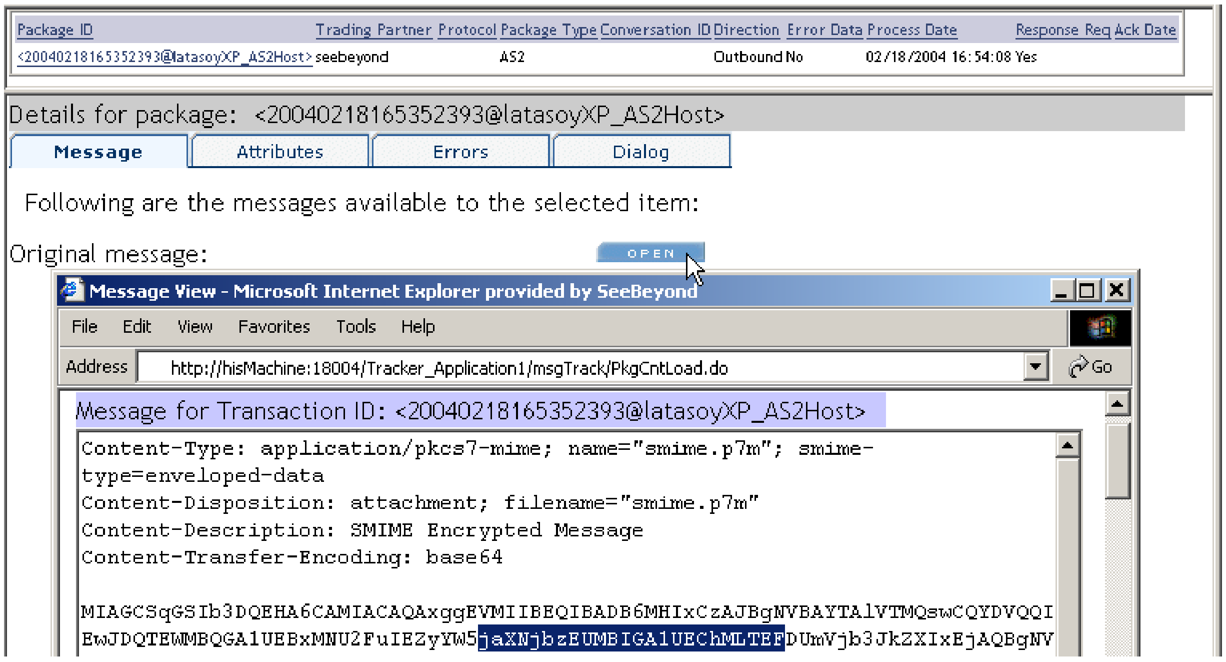
Purpose: On a package-by-package basis, you can examine the message text.

1 After obtaining results from a search using any of the procedures mentioned earlier, click the package ID for any of the returned results.

2 In the "Details for package <*package-ID*>" pane, click **Open** to see the contents (possibly encrypted) of the original message.

Result: See Figure 71. You can use cut, copy, and paste on any text in the window.

Figure 71 Message Tracking, Showing Package Details and Message Content



Package ID	Trading Partner	Protocol	Package Type	Conversation ID	Direction	Error Data	Process Date	Response Req	Ack Date
<20040218165352393@latasoyXP_AS2Host>	seebeyond	AS2			Outbound	No	02/18/2004 16:54:08	Yes	

Details for package: <20040218165352393@latasoyXP_AS2Host>

Following are the messages available to the selected item:

Original message:

Message View - Microsoft Internet Explorer provided by SeeBeyond

Address http://hisMachine:18004/Tracker_Application1/msgTrack/PkgCntLoad.do

Message for Transaction ID: <20040218165352393@latasoyXP_AS2Host>

```
Content-Type: application/pkcs7-mime; name="smime.p7m"; smime-
type=enveloped-data
Content-Disposition: attachment; filename="smime.p7m"
Content-Description: SMIME Encrypted Message
Content-Transfer-Encoding: base64

MIAGCSqGSIb3DQEHA6CAMIACAQAxggEVMIIBEQIBADB6MHIxCzAJBgNVBAYTAlVTMQswCQYDVQQI
EwJDQTEWMBQGA1UEBxMNU2FuIEZyYW5jaXNjbzEUMBIGA1UEChMLTEFDUmVjb3JkZXIxEjAQBgNV
```

Chapter 9

Implementation Scenario: CMScheduler

eXchange Integrator includes a complete sample implementation, included in the **eXchangeDocs.sar** file, that allow you to see the end results without having to go through all the design-time steps. This chapter provides a sample scenario showing how eXchange can be used to achieve B2B solutions without using add-on protocols.

The steps for the sample implementation occur in four phases:

Initial Setup Steps	In these steps, you ensure that prerequisites are met, obtain the necessary sample materials, extract sample files, and import the sample projects. See **section 9.1 on page 127**.
Design Steps in Enterprise Designer	In these steps, you use Enterprise Designer to add and configure externals, view components in the B2B host project, and it, creating an eXchange Service for the B2B host. Then you view the components in the main project and activate it. See **section 9.2 on page 129**.
Design Steps in ePM	In these steps, you use the eXchange Partner Management (ePM) facility to create a trading partner, view and specify parameter values for it, and activate it. See **section 9.3 on page 134**.
Runtime Steps	In these steps, you bootstrap two logical hosts, apply the activated deployments, and follow the progress as the B2B protocol is triggered by the ChannelManager and messages are written out to the Batch eWay. See **section 9.4 on page 140**.

Overview of the Sample Implementation

The sample implementation uses a Channel Manager in Scheduler mode to periodically “wake up” and trigger a B2B protocol process whose output is written to a Batch eWay.

9.1 Initial Setup Steps

In this section

- **Installing the Sample Files for CMScheduler on page 128**
- **Setting Up the Sample Environment on page 129**
- **Importing the Sample Projects on page 128**

9.1.1 Installing the Sample Files for CMScheduler

These steps assume the existence of a temporary eXchange directory for sample files, such as **C:\temp\eXchange\**. You will extract the sample files to this directory so that you can conveniently access the files in later procedures.

To install the sample files

Before you begin: Your repository must already be running, and you must be logged in to Enterprise Manager. If you have already uploaded the documentation for eXchange, you can skip steps 1 and 2 and start with step 3.

1. In the ADMIN tab, if you have not already done so, browse to the [...]\Documentation**ProductsManifest.xml** file and submit it.
2. In the ADMIN tab, if you have not previously done so, browse to the **eXchangeDocs.sar** file, select it, and click the **upload now** button.
3. In the DOCUMENTATION tab, under Products, click **eXchange Integrator**
4. In the window that appears on the right side, click **Download Sample**
5. Preserving file paths, extract the contents to your **C:\temp\eXchange\** directory.

Result: The following directories and files are created under **C:\temp\eXchange\**:

```
Sample\CMScheduler\Projects\eXCMScheduler.zip
Sample\CMScheduler\TradingPartners\CMScheduler_TP.xml
```

9.1.2 Importing the Sample Projects

To import the sample projects and their template folders

Before you begin: Your repository must already be running, and you must be logged in to Enterprise Designer. If your repository already has a project at the root level named CM_Scheduler_Host or CM_Scheduler_Protocol, delete or rename it.

1. In Project Explorer, after saving any work in progress, right-click the repository and, on the popup menu, click: **Import**
2. In the **Import Manager** dialog, browse to the folder where you installed the sample project (for example, C:\temp\eXchange\CMScheduler\Projects), select **eXCMScheduler.zip**, and then click Open.
3. Click the **Import** button to import both items (two projects).
4. When the import finishes, click OK to clear the confirmation, and then click Close.

Result: Two new project folders (**CM_Scheduler_Host** and **CM_Scheduler_Protocol**) are added to the repository.

9.2 Design Steps in Enterprise Designer

For the sample implementation, design-time steps in Enterprise Designer consist of the following:

- **Setting Up the Sample Environment on page 129**
- **Viewing and Activating the B2B Host Project on page 131**

9.2.1 Setting Up the Sample Environment

The sample assumes you will use default configurations for all servers where possible, and that you will make any changes where needed. For example: If you use anything other than a SeeBeyond Integration Server on ports 18000–18009, make adjustments in step 3 (ports) and/or step 4 (type of Integration Server).

To create the sample environments

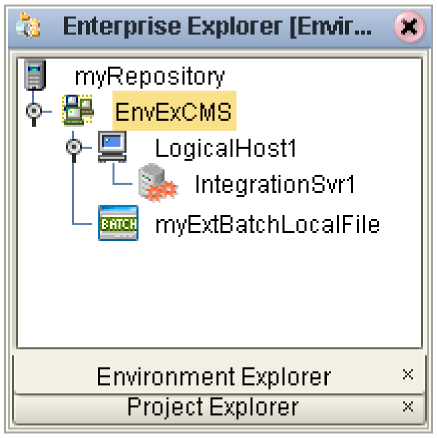
1. In Enterprise Designer, near the lower left of the window, click the **Environment Explorer** tab.
2. In the Environment Explorer tree, right-click the repository and, on the popup context menu, click **New Environment**
   - Rename the newly created environment to **EnvExCMS**
3. Right-click EnvExCMS and, on the menu, click: **New Logical Host**
   - Retain the default name: LogicalHost1

Tip: *For a second or subsequent logical host: Check it out if necessary; then, right-click it and open its properties; click **Logical Host Configuration** and change the value **Logical Host Base Port** to a larger multiple of 1000 (19000 if ports 19000-19009 are unused; otherwise 20000, or 21000); finally, close the properties sheet.*

4. Right-click LogicalHost1 and click: **New SeeBeyond Integration Server**
   - Retain the default name: IntegrationSvr1
5. Right-click EnvExCMS > **New BatchLocalFile External System**
   - Name it **myExtBatchLocalFile** and click OK.
6. On the main toolbar, click **Save All**.

Result: The environment, named EnvExCMS, now has all but two of the externals needed by the projects. Steps for the outbound Oracle external are provided in the following procedure, and the final external will be created by activating the project containing the B2B host.

Figure 72 Sample Environment, Before Configuration



To create and configure the Oracle external

Before you begin: Your eXchange 5.0.4 Oracle database must be accessible, and you must you know its SID, username, and password.

1. In the Environment Explorer tree, right-click EnvExCMS and, on the popup context menu, click **New Oracle External System**
   - Name it **myExtOracleOut**, designate it **Outbound**, and click OK.
2. Right-click myExtOracleOut and configure properties appropriately. For example:
   - **DatabaseName:** *exch50 (change this to the SID for your eXchange Oracle database)*
   - DataSourceName: **local**
   - **Password:** *(replace this with the password for your eXchange 5.0.4 database user)*
   - **PortNumber: 1521** *(change this only if your Oracle administrator changed the default)*
   - **ServerName:** *myMachine (change this to the hostname of the Oracle server machine)*
   - **User:** *ex504Adm (change this to the username for your eXchange database user)*
3. When all properties have been configured correctly for your site, click OK.
4. Collapse the EnvExCMS tree, click **Save All**, and close all canvases.

Result: **EnvExCMS** now has all but one of the externals needed by the projects.

The final external needed for this environment, an eXchange service, will be created by activating the project that contains the B2B host.

9.2.2 Viewing and Activating the B2B Host Project

In the Project Explorer tree, open the B2B host project (named **CM_Scheduler_Host**) to display its components. This is a quick guide to the B2B host's contents. Activating this project will create an eXchange service that acts as a channel manager and provides a connection to the eXchange database.

Components of the B2B host project

- **myMsgService** is the only message service used by the B2B host:
  - It contains a single outbound messaging action: It originates from the internal system, passes through the B2B host, and arrives at the external trading partner.
  - The (outbound) internal messaging action is named **internalOutbound**; the (outbound) external messaging action is named **externalOutbound**.
  - Its messaging attributes definition (MAD) is EmptyMAD.
- **EmptyMAD** is a blank custom messaging attributes definition.
- **ChannelManagerTAD** is the transport attributes definition for Channel Manager. It uses ChannelManagerMode of SCHEDULER, polling every 5000 milliseconds and writing to the output path and filename to be specified by the trading partner.
- **ChannelManagerTAD_OTD** is the OTD generated from ChannelManagerTAD.
- **CMS_Host** is the B2B host itself:
  - **Business Protocols**—Only one service is referenced (under EmptyMAD): myMsgService
  - **External Delivery Channels**—Only one external delivery channel is defined: **EmptyMAD_deliverychannel**. For transport to and from trading partners, this channel references the standard SeeBeyond-supplied FILE transport attributes definitions.
  - **Internal Delivery Channels**—Only one internal delivery channel is defined: **Internal_Delivery_Channel1**. For transport, this channel references the ChannelManagerTAD that was defined for this project, in only one direction (Receiver).
- **CMS_Host_CMap** is the map whose activation will create the eXchange service:
  - Its input is an instance of CMS_Host, with two outbound connections.
  - Its only output is an instance of Oracle, with two inbound connections.
  - Connecting to both is an instance of a SeeBeyond-supplied tracking application.

To activate the B2B host, creating the eXchange service

Before you begin: Your environment must contain a well-configured Oracle external (see the preceding procedure), and the environment must be named **EnvExCMS**— that is, it must correspond to the name of your host project.

1. Right-click CMS_Host and, on the popup context menu, point at **New** and click **Deployment Profile**
2. Keep the default name (Deployment1), point it at **EnvExCMS**, and click OK.

   The Deployment Editor opens. Its left pane has two services and two Oracle eWays.

3 On the right side, minimize all windows except LogicalHost1 and myExtOracleOut.

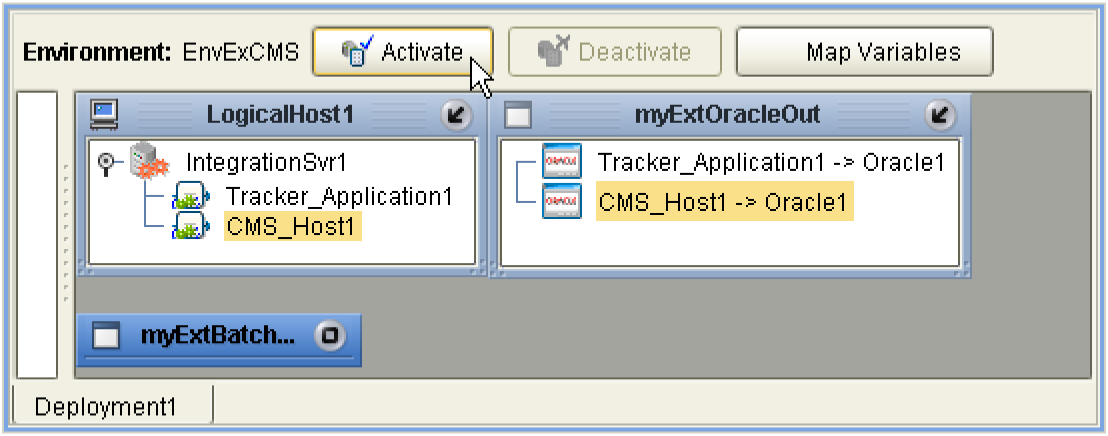
4 One by one, drag the two services into LogicalHost1 and under **IntegrationSvr1**.

5 One by one, drag the two Oracle eWays into **myExtOracleOut**. See Figure 73.

Tip: *If myExtOracleOut refuses to accept eWays, it may be an indication of:*

- *The Oracle database instance it references is inaccessible. Ensure it is running and that the myExtOracleOut properties match its hostname, SID, username, and password. If necessary, see* **"To create and configure the Oracle external" on page 130**.
- *It was misdefined as inbound. Delete myExtOracleOut and re-create it as outbound. Then: Click* ***Save All****, followed by* ***Refresh All from Repository****.*

Figure 73 Deployment Profile for B2B Host, Before Activation



6 Click **Save All**, and then click **Activate**.

7 In response to the dialog box, click **No**; that is, do *not* apply to Logical Host(s).

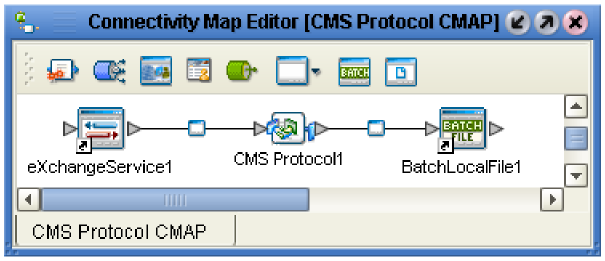
Result: A new external is created, named **CMS_Host1 eXchange Service**. The projects now have all the externals they need. Save all of your work, close all canvases, and click **Refresh All from Repository**.

9.2.3. Introducing the Project: CM_Scheduler_Protocol

The project itself is quite simple, consisting of nothing more than the following.

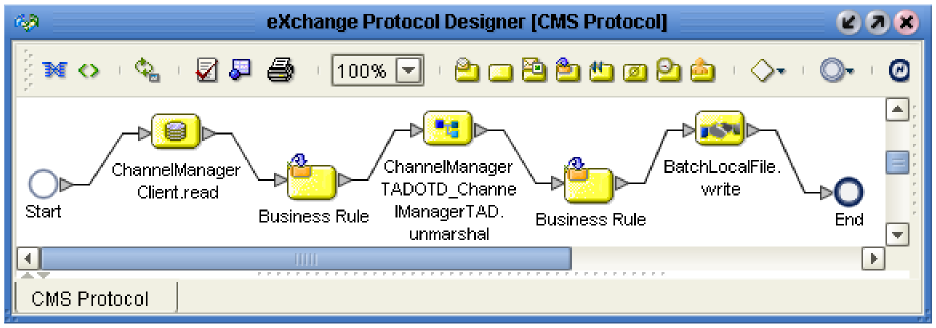
- **CMS Protocol CMAP**—A very simple connectivity map whereby the eXchange Service that you built in the previous procedure is connected to the custom-built B2B protocol process mentioned above, which is connected to a BatchLocalFile external system.

Figure 74 Connectivity Map for Sample CM_Scheduler_Protocol



- **CMS Protocol**—A simple custom-built B2B protocol process (supplied in the sample project) that performs a ChannelManager **read**, followed by an **unmarshal** (using the OTD that was generated from the custom ChannelManager transport attributes definition), and finally a BatchLocalFile **write**.

Figure 75 B2B Protocol Process for Sample CM_Scheduler_Protocol


eXchange Protocol Designer [CMS Protocol]
100%
Start
ChannelManager Client.read
Business Rule
ChannelManager TADOTD_ChannelManagerTAD. unmarshal
Business Rule
BatchLocalFile. write
End
CMS Protocol

9.2.4. Creating and Activating the Project Deployment Profiles

To activate the main CMScheduler project

Before you begin: Your environment must be named **EnvExCMS** (that is, it must correspond to the name of your host project), and it must contain an eXchangeService; if necessary, see the **procedure on page 131**.

1. In the Project Explorer tree, right-click **CM_Scheduler_Protocol** and, on the popup context menu, point at New and click **Deployment Profile**
2. Keep the default name (Deployment1), point it at **EnvExCMS**, and click OK.

   The Deployment Editor's left pane displays: A service for the B2B protocol process (**CMS Protocol1**); an outbound eWay for BatchLocalFile; and an eWay for an outbound eXchangeService.

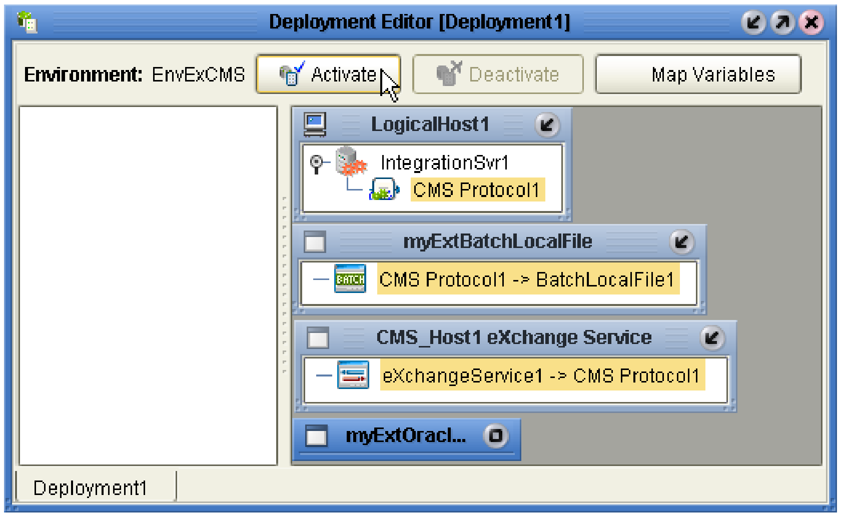
3 Drag the service into LogicalHost1 and under **IntegrationSvr1**.

4 Drag the outbound BatchLocalFile eWay into **myExtBatchLocalFile**.

5 Drag the eXchangeService eWay into **CMS_Host1 Exchange Service**. When you drag and drop the eXchangeService eWay, specify **EmptyMAD** as the protocol.

6 When Deployment1 is complete—that is, when all components in the CM_Scheduler_Protocol project are associated with corresponding servers—click **Activate**. See Figure 76.

Figure 76 Activated Deployment Profile for CM_Scheduler_Protocol



7 In response to the dialog box, click **No**; that is, do *not* apply to Logical Host(s).

9.3 Design Steps in ePM

For the sample implementation for CMScheduler, design-time steps in eXchange Partner Manager (ePM) consist of the following:

- **Importing Trading Partners on page 134**
- **Configuring Trading Partner Parameters on page 136**
- **Activating the Trading Partner on page 139**

9.3.1 Importing Trading Partners

Before you begin: Your repository and your eXchange 5.0.4 Oracle database must be running and accessible, and you must have completed the activation steps in the previous section. Enterprise Designer does not need to be running, and you do not need to have any logical hosts running.

To start eXchange Partner Manager (ePM)

1 Start a *new* browser session (that is, do ***not*** clone a new window of an existing session) pointing it at a repository URL, with **epm** appended. For example:

- If your repository were running local on port 12000, the URL would be:

  ```
  http://localhost:12000/epm
  ```

- For a repository running on machine herMachine on port 33000, it would be:

  ```
  http://herMachine:33000/epm
  ```

- As usual, IP addresses are also permissible:

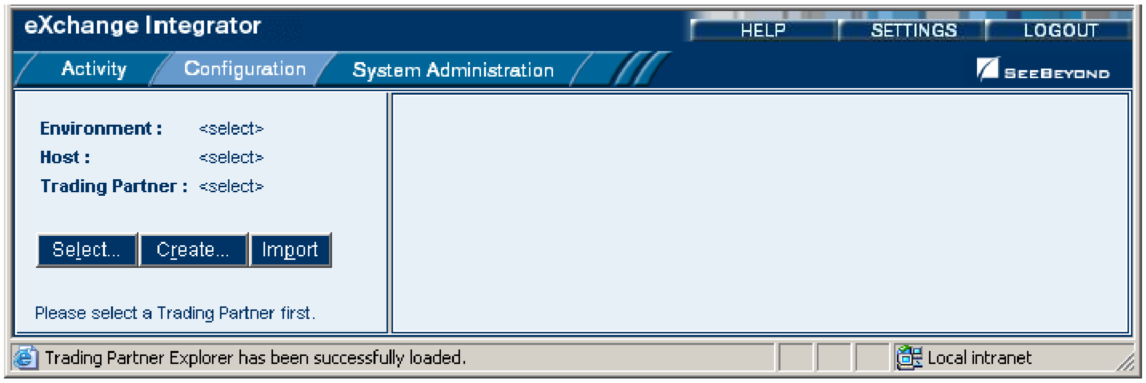
  ```
  http://10.18.75.85:36271/epm
  ```

The string **epm** is case sensitive. In other words: ePM, Epm, and EPM are all errors.

2 When the sign-in screen appears, enter the Enterprise Manager username and password if necessary and click **Sign In**.

Result: The status bar (along the lower margin of the window) confirms that Trading Partner Explorer has loaded successfully, and the initial ePM screen appears, with no environment, host, or trading partner. See Figure 77.

Figure 77 Initial ePM Screen



Environment : <select>
Host : <select>
Trading Partner : <select>

Please select a Trading Partner first.

Trading Partner Explorer has been successfully loaded.

To import trading partner myTP1 into EnvExCMS

1 From the initial ePM window, in the upper left side, click **Import**.

A new window opens (**Import a New Trading Partner**), prompting you to select a B2B host and specify a name for the trading partner.

2 Open the B2B Repository and **EnvExCMS** and click **CMS_Host1**.

3 Enter **myTP1**, browse to C:\temp\eXchange\Sample\CMScheduler and open **CMScheduler_TP.xml**, and then click **Import**.

Result: In the explorer tree, under EnvExCMS, new trading partner **myTP1** appears.

To import trading partner myTP2 into EnvExCMS

1 In the explorer (left) pane of the ePM window, click **Import** again.

2 As before, open the B2B Repository and **EnvExCMS** and click **CMS_Host1**.

3 Enter **myTP2**, browse to C:\temp\eXchange\Sample\CMScheduler and open **CMScheduler_TP.xml**, and then click **Import**.

Result: The explorer tree displays **myTP2** under EnvExCMS. However, the previous trading partner has not been lost or overwritten.

To find trading partner myTP1

1 In the upper left side of the explorer pane, click **Select**.

  A new window opens, prompting you to select a B2B host and specify a search string for trading partner.

2 Open the B2B Repository and **EnvExCMS** and click **CMS_Host1**.

3 Click **Search**, and then, from the list, click **myTP1**.

Result: In the explorer tree, under EnvExCMS, trading partner **myTP1** reappears.

9.3.2 Configuring Trading Partner Parameters

When you imported the trading partner, parameter settings were valuated in part based on parameters stored in the export file, and in part based on the name of the trading partner. In this section, you will set or update the following:

- **Parameters for the Delivery Channel on page 136**
- **Parameters for the Internal Delivery Channels on page 137**

Parameters for the Delivery Channel

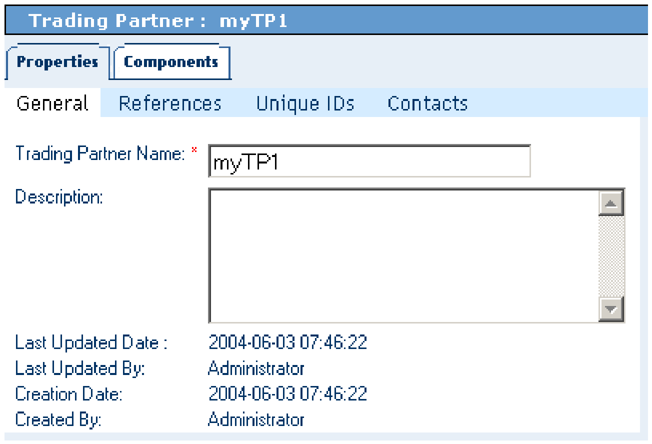
Purpose: To set the parameters governing the B2B host's message exchange with **myTP1**. You are configuring a trading partner for the EnvExCMS environment, and so take the viewpoint of the CMS B2B host: "ToPartner" means "to myTP1"; "FromPartner" means "from myTP1".

To configure the delivery channel parameters for trading partner myTP1

1 In the explorer (lower left) side of the ePM screen, click **myTP1**.

  The canvas displays the trading partner's general properties. See Figure 78.

Figure 78 Trading Partner myTP1: General Properties



Trading Partner Name: * myTP1
Description:
Last Updated Date : 2004-06-03 07:46:22
Last Updated By: Administrator
Creation Date: 2004-06-03 07:46:22
Created By: Administrator

2 Click the **Components** tab.

The trading partner's delivery channel parameters are displayed. See Table 37.

Table 37 Delivery Channel Parameters for myTP1

Binding Name	EmptyMAD_delivery_channel1
ToPartner Transport Name	FILE
FromPartner Transport Name	FILE
Packager Name	EmptyMAD

3 Click the binding name, **EmptyMAD_delivery_channel1**.

The delivery channel's general properties are displayed.

4 Click the **ToPartnerTransport** tab and edit the values for FilePattern and Directory so that they point to an appropriate file mask in an appropriate location.

5 When you are done, click **Save**. (This required, even if no values are changed.)

6 Click the **FromPartnerTransport** tab and edit the values for FilePattern and Directory so that they point to an appropriate file mask in an appropriate location.

7 When you are done, click **Save**. (This required, even if no values are changed.)

8 Click the **ToPartnerPackaging** tab and click **Save**. (This is always required, even if no values are entered.)

9 Click the **FromPartnerPackaging** tab and click **Save**. (This is always required, even if no values are entered.)

Result: For trading partner **myTP1**, the parameters for external delivery channel **EmptyMAD_delivery_channel1** are now set correctly.

Parameters for the Internal Delivery Channels

Purpose: To set the parameters governing the B2B host's message processing when handling messages received internally that are destined for **myTP1**. (You take the viewpoint of the CMS B2B host: "Receiver" and "fromInternal" both mean "received from the internal system by the B2B host".)

To configure the internal delivery channel parameters for trading partner myTP1

1 With trading partner **myTP1** active in the tree (left pane) and the Components tab active in the canvas (right pane) of the ePM window, click the **Internal Delivery Channels** subtab.

The canvas displays the single internal delivery channel associated with this trading partner. See Table 38.

Table 38 Internal Delivery Channel Parameters for Trading Partner myTP1

Binding Name	Internal_Delivery_Channel1
Direction	Receiver
Transport Name	ChannelManagerTAD

2 Click the binding name, **Internal_Delivery_Channel1**.

The internal delivery channel's general properties are displayed.

Click the **Receiver Transport** tab and edit its parameters as shown in Table 39.

Table 39 Receiver Transport Parameters for Trading Partner myTP1

Output File Directory	C:\temp\eXchange\Sample\Data
Output File Name	myTP1_CMS_OutputFile_%d_%H_%m_%s
PollMilliSeconds	15000
ChannelManagerMode	SCHEDULER

3 Click **Save**. (This required, even if no values are changed.)

Result: For trading partner **myTP1**, the parameters for the internal delivery channel are now set correctly.

Parameters for the Messaging Actions

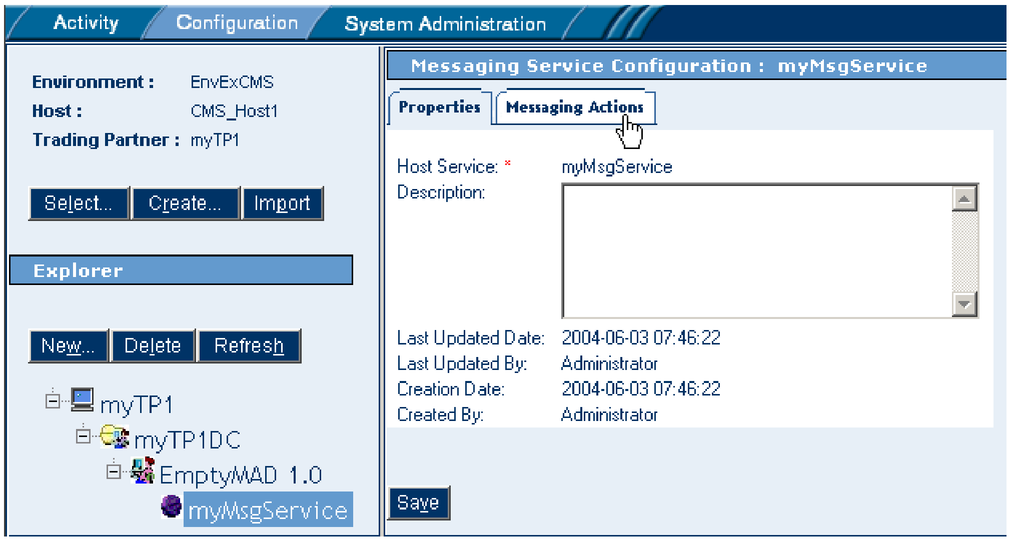
Purpose: To associate each messaging action with the correct external and internal delivery channel and set other parameters if necessary. The messaging actions are defined by the B2B host's messaging service.

To configure the messaging actions for trading partner myTP1

1 In the explorer (lower left) side of the ePM screen, click **myTP1**, open its profile (myTP1DC) and messaging protocol (EmptyMAD 1.0), and click its associated messaging service: **myMsgService**

The canvas displays the messaging service's general properties. See Figure 79.

Figure 79 Messaging Service myMsgService: General Properties



Environment : EnvExCMS
Host : CMS_Host1
Trading Partner : myTP1

Messaging Service Configuration : myMsgService

Host Service: * myMsgService
Description:
Last Updated Date: 2004-06-03 07:46:22
Last Updated By: Administrator
Creation Date: 2004-06-03 07:46:22
Created By: Administrator

2 Click the **Messaging Actions** tab.

The canvas displays the single messaging action of this service: **externalOutbound**.

3 Open the messaging action and verify parameters as shown in Table 40.

Table 40 Messaging Action Parameters for externalOutbound

Send To Partner?	true
Delivery Channel	EmptyMAD_delivery_channel1
Internal Delivery Channel	Internal_Delivery_Channel1
Mime Configuration	[None]

Note: *Internal Delivery Channel bindings are always required for a project to run. Although it is possible to activate a trading partner whose messaging actions lack IDCs, the Channel Manager read operation would never be triggered.*

4 Click **Save**. (This required, even if no values are changed.)

Result: For trading partner **myTP1**, all parameters are now set correctly.

9.3.3 Activating the Trading Partner

To activate trading partner myTP1

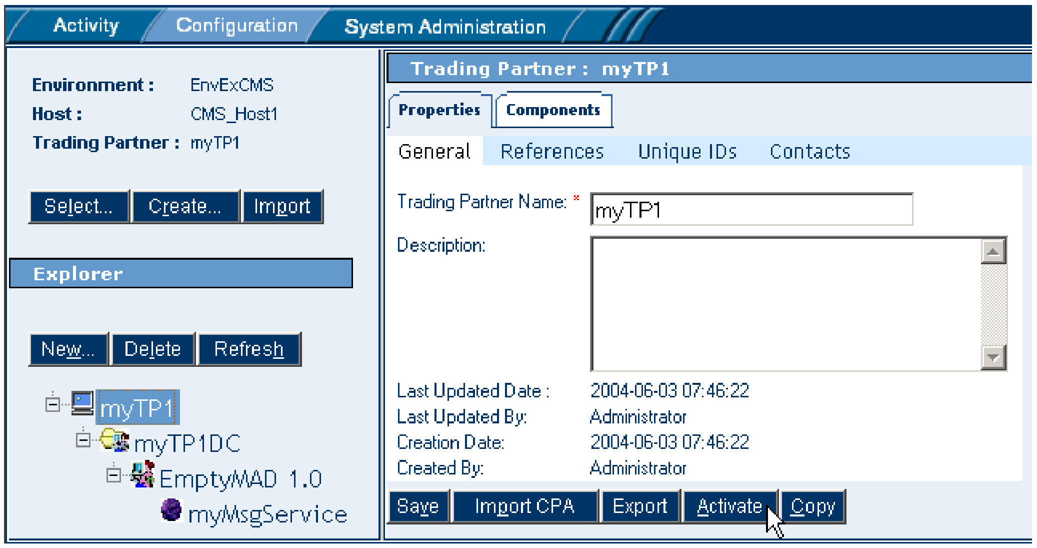
Purpose: To save all the configuration information to the Oracle database to make it available at run time.

Before you begin: Your eXchange 5.0.4 Oracle database for the corresponding B2B host must be running.

1 In the ePM explorer tree (left pane), click **myTP1**.

2 In the bottom center of the canvas, click the **Activate** button. See Figure 80.

Figure 80 Activating Trading Partner myTP1



Environment : EnvExCMS
Host : CMS_Host1
Trading Partner : myTP1

myTP1
myTP1DC
EmptyMAD 1.0
myMsgService

Trading Partner Name: * myTP1
Description:

Last Updated Date : 2004-06-03 07:46:22
Last Updated By: Administrator
Creation Date: 2004-06-03 07:46:22
Created By: Administrator

3 In response to the confirmation prompt, click **Activate**.

The canvas displays a confirmation: **Trading Partner is successfully activated.**

Result: Trading partner **myTP1** is entirely complete and ready to be run. (However: If a logical host is already running, these changes are not made to it until you either reactivate the project or right-click the logical host and click **Apply**.)

9.3.4 Finding, Configuring, and Activating Other Trading Partners

Earlier, in **section 9.3.1 on page 134**, you imported trading partner **myTP2** for this environment and host, but you have not yet configured or activated it.

To find trading partner myTP2

1 In the upper left side of the explorer pane, click **Select**.

A new window opens, prompting you to select a B2B host and specify a search string for trading partner.

2 Open the B2B Repository and **EnvExCMS** and click **CMS_Host1**.

3 Click **Search**, and then, from the list, click **myTP2**.

Result: In the explorer tree, under EnvExCMS, trading partner **myTP2** reappears.

For trading partner **myTP2**, repeat the same configuration and activation procedures you just followed for myTP1, making the appropriate changes:

- **Configuring Trading Partner Parameters on page 136**
- **Activating the Trading Partner on page 139**

Result: After activation, when the canvas displays the confirmation message for myTP2 (**Trading Partner is successfully activated**), both trading partners are entirely complete and ready to be run. (However: If a logical host is already running, these changes are not made to it until you either reactivate the project or right-click the logical host and click **Apply**.)

9.4 Runtime Steps

9.4.1 Starting the Logical Hosts

These steps assume you have already installed two or more logical hosts.

To bootstrap the logical hosts

1 Open a command prompt and change directories to the location of your logical host's bootstrap executables. For example:

```
cd <logicalhostA>\bootstrap\bin
```

2 Start the bootstrap script using appropriate parameters. For example:

```
bootstrap -r http://myBox:12345/myRepository -i myId -p myPassword
-e EnvExCMS -l LogicalHost1
```

- For the **-r** (repository) parameter), supply the correct URL with repository name.
- For the **-i** and **-p** (ID and password) parameters, supply the appropriate values.
- For **-e** (environment) parameter, use: **EnvExCMS**
- For **-l** (logical host name) parameters, use: **LogicalHost1**

Result: After a time, the logical host starts running, and all activated projects that reference EnvExCMS are automatically applied to it.

3 Repeat steps 1 and 2 on a different logical host, referencing the same repository but pointing it at the EnvExCMS environment. For example:

```
cd <logicalhostB>\bootstrap\bin
bootstrap -r [...] -e EnvExCMS -l LogicalHost1
```

To apply environment changes when a logical host is running

Purpose: If changes are made to parameters in an environment component while a logical host is running, use these steps to apply the changes without having to shutdown and re-bootstrap the logical host.

1 In Enterprise Designer, in Environment Explorer, open the environment where the changes have occurred.

2 Right-click the logical host that is running and, on the popup menu, click **Apply**

3 Repeat the previous step as needed for other logical hosts in the same environment.

Result: In the back end, a "mini-shutdown/mini-rebootstrap" occurs, and the changes are applied to the running logical host.

Chapter 10

Troubleshooting eXchange

10.1 Identifying and Resolving Problems

The **Readme.txt** file contains a list of known issues for eXchange and other products. It is available both on the installation media, or via the DOCUMENTATION tab of Enterprise Manager, or in <*ican50*>\repository\server\webapps\ICANDocs\.

For help resolving some commonly encountered difficulties, see Table 41.

Table 41 Troubleshooting: Symptoms, Causes, and Remedies

Symptom	Possible Causes	Suggested Remedies
In Enterprise Manager, while uploading, receive message "installation failed"; and/or log says "Could not install: <*product*>"	One or more prerequisite **.sar** files have not been installed.	Ensure you are licensed to install the product. Install its prerequisite **.sar** files (see **section 3.2.1 on page 12**) one by one and then upload the product again.
Messages in log files or on the console are too sparse or too abundant.	The level of message output in the log4j.properties file is not set appropriately.	Depending on the module that is too terse or too verbose, edit the file in one or more of these <*ican50*> locations: ▪ edesigner\bin\log4j.properties ▪ ican50\ESRs\log4j.properties ▪ monitor\config\log4j.properties ▪ repository\server\conf\log4j.properties
In the Deployment Editor, cannot drop an Oracle eWay onto the Oracle external.	▪ The eWay is Inbound and the external Outbound, or vice versa. ▪ The Oracle host is not configured to match a running Oracle service. ▪ The Oracle service is not running.	▪ Reconfigure the eWay (in the connectivity map) or the Oracle host (in the environment editor). ▪ In the environment editor, double-check the properties of the Oracle host. ▪ Start the Oracle service (e.g., using **Control Panel >Admin... > Services**).
Upon running bootstrap, receive message "An instance of bootstrap is already running in this directory."	A previous bootstrap was shut down improperly.	First, try running the **shutdown** script with the **-c** flag. If the problem continues to persist after this, find and remove the **bootstrap.lock** file.

Table 41 Troubleshooting: Symptoms, Causes, and Remedies (Continued)

Symptom	Possible Causes	Suggested Remedies
Any of the following runtime messages are received: ▪ Error while decrypting. ▪ RSA_ not supported ▪ Error while encrypting in PKCS7 format. ▪ java.lang.SecurityException	Incorrect versions of the **US_export_policy.jar** and **local_policy.jar** files are in use.	See **section 3.4 on page 36** for how to obtain the correct versions. If you have installed the correct versions and still get this message, then run **shutdown**; delete all subdirectories under <*logicalhost*>, except **bootstrap** and **jre**; and then re-bootstrap to regenerate the keystores.
Changes made to the design, environment, or trading partner profile are not picked up at run time. For example: ▪ A component is added, but ignored. ▪ A configuration error is corrected, but the runtime behavior is unchanged. ▪ A keystore is changed, but the new password is not recognized.	Stale information is being retained by the repository or logical host or eXchange database.	▪ Design changes: Double-check the connections between components; save all changes and check in all components; exit all canvases; refresh all from repository; reopen and reactivate the deployment profile. ▪ Environment changes: Either right-click the logical host and click Apply, or select the Apply Changes checkbox when reactivating the deployment profile, or use the **-f** flag when rebootstrapping. ▪ TP profile changes: In ePM, go through the TP's parameters and save each one; then reactivate the trading partner.
Message Tracking reports "No messages found".	▪ The statement is accurate, and no messages have been processed. ▪ The wrong host is specified and/or the filters are too stringent.	▪ Look inside the eXchange database: If the EX_MSG_CONTENT table has no messages, the statement is accurate; ensure that the ChannelManager **track** operation is being used correctly. ▪ Be sure the correct host is selected, and set the Search parameters to **ALL** for both Trading Partner and Protocol.
The B2B protocol starts, and ChannelManager retrieves the delivery channel profile, but nothing happens.	▪ There is no input being staged. ▪ There is no binding for the internal delivery channel to the location where input exists.	▪ Double-check the values for input directory and file mask (and, for FTP, hostname and login information). Ensure that input data exists. ▪ In the TP profile, in Messaging Service Configuration, set the internal delivery channel for each messaging action so that it points at the correct input data location. Then, if necessary, reactivate.
Unexpected errors occur at runtime, with the following error message text: "java.sql.SQLException: ORA-01000: maximum open cursors exceeded"	The load on the eXchange Oracle database is too great for the default settings to handle.	In the **init.ora** file for the eXchange database, increase the value for the **open_cursors** parameter to 500. See **"Modifying the init.ora File for the eXchange Database" on page 31**.

Appendix A

Method Palette

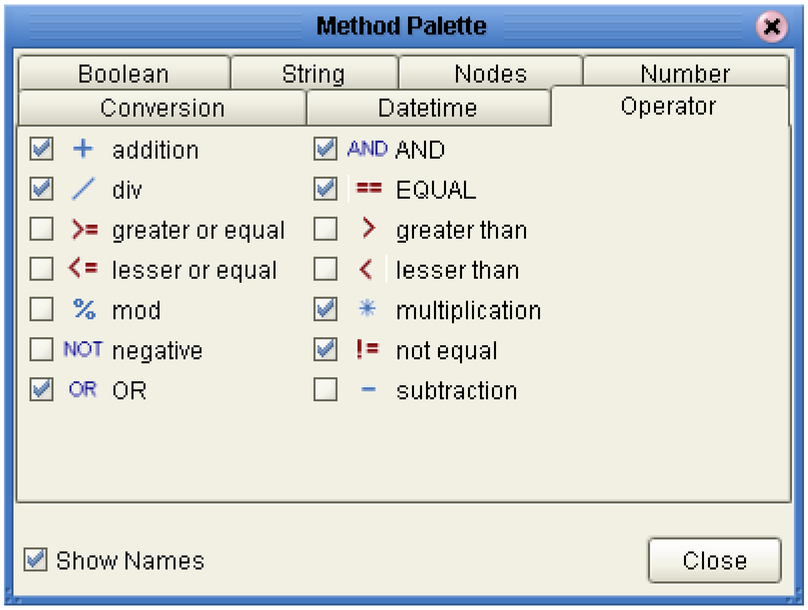
In this appendix:

This appendix describes each method that appears in the Method Palette of the eXchange Protocol Designer.

A.1 Operators

Operators are the methods that allow you to manipulate data with standard mathematical operators.

Figure 81 Method Palette: Operator Tab



☑ + addition	☑ AND AND
☑ / div	☑ == EQUAL
☐ >= greater or equal	☐ > greater than
☐ <= lesser or equal	☐ < lesser than
☐ % mod	☑ * multiplication
☐ NOT negative	☑ != not equal
☑ OR OR	☐ - subtraction

☑ Show Names

Table 42 Operator Methods

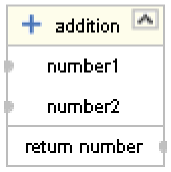
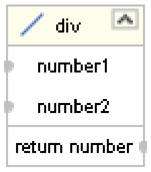
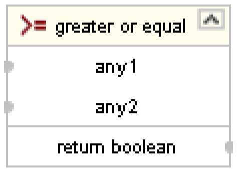
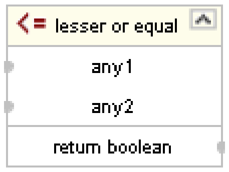
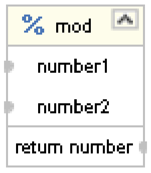
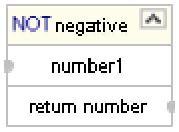
Symbol	Name	Function
	addition	Adds the value of *number1* to the value of *number2*, returns the sum.
	div	Divides the value of *number1* by the value of *number2*, returns the quotient.
	greater or equal	Returns Boolean true if *number1* is greater than or equal to *number2*; otherwise, returns Boolean false.
	lesser or equal	Returns Boolean true if *number1* is less than or equal to *number2*; otherwise, returns Boolean false.
	mod	Used to divide two numbers and return only the remainder.
	negative	Converts the input number to negative. Result is a negative number having the same absolute value as the input number.

Table 42 Operator Methods (Continued)

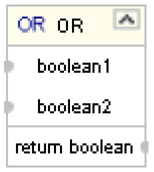
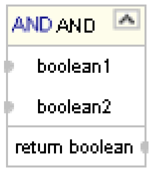
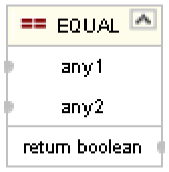
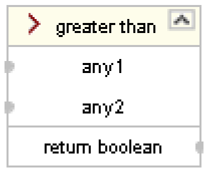
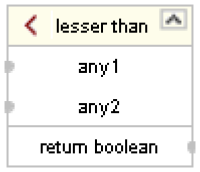
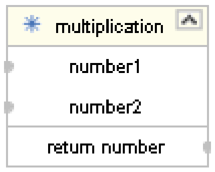
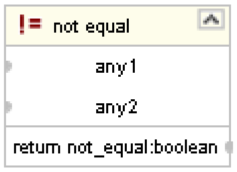
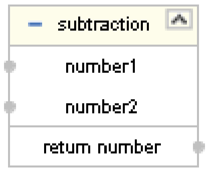
Symbol	Name	Function
	OR	Returns Boolean false if both *boolean1* and *boolean2* are false; otherwise, returns Boolean true.
	AND	Returns Boolean true if both *boolean1* and *boolean2* are true; otherwise, returns Boolean false.
	EQUAL	Returns Boolean true if *number1* is equal to *number2*; otherwise, returns Boolean false.
	greater than	Returns Boolean true if *number1* is greater than *number2*; otherwise, returns Boolean false.
	lesser than	Returns Boolean true if *number1* is less than *number2*; otherwise, returns Boolean false.
	multiplication	Multiplies the value of *number1* by the value of *number2*, returns the product.

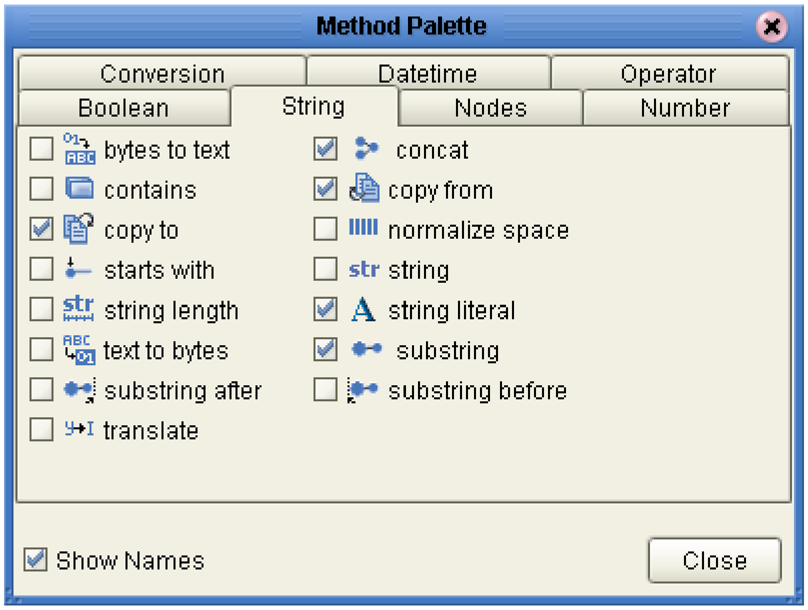
Table 42 Operator Methods (Continued)

Symbol	Name	Function
	not equal	Returns Boolean true if *number1* is not equal to *number2*; otherwise, returns Boolean false.
	subtraction	Subtracts the numerical value of *number2* from the numerical value of *number1*, returns the difference.

A.2 String

The String methods allow you to manipulate string data.

Figure 82 Method Palette: String Tab



Method Palette

Tabs: Conversion, Datetime, Operator, Boolean, String, Nodes, Number

☐ bytes to text	☑ concat
☐ contains	☑ copy from
☑ copy to	☐ normalize space
☐ starts with	☐ string
☐ string length	☑ string literal
☐ text to bytes	☑ substring
☐ substring after	☐ substring before
☐ translate	

☑ Show Names

Close

Table 43 String Methods

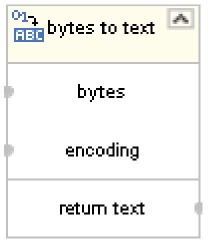
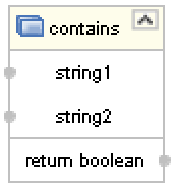
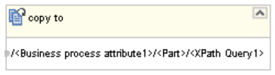
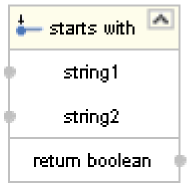
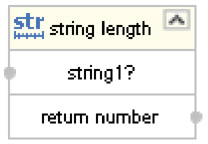
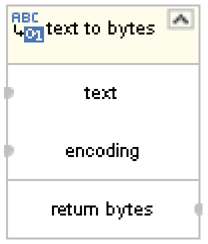
Symbol	Name	Function
	bytes to text	Decodes bytes into text using the specified encoding. If no encoding is specified, the platform's default encoding is used.
	contains	Returns true if the second string is contained within the first string, otherwise it returns false
	copy to	Allows you to type in the xpath expression for the destination of a copy operation. This is useful for entering xpath predicates. Note: This is for advanced users who are familiar with xpath and BPEL syntax.
	starts with	Returns true if the first string starts with the second string, otherwise it returns false
	string length	Returns the number of characters in a string
	text to bytes	Encodes the input text into a sequence of bytes using the specified encoding. If no encoding is specified, the platform's default encoding is used

Table 43 String Methods (Continued)

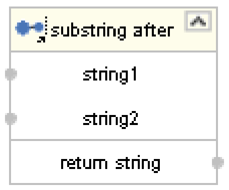
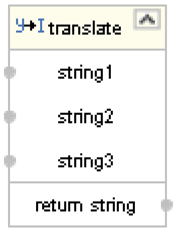
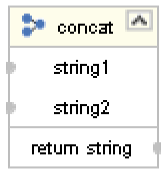
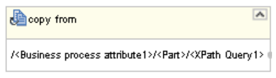
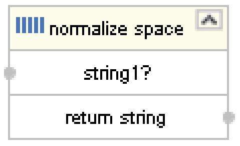
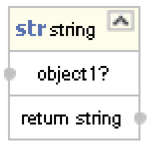
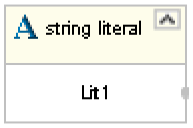
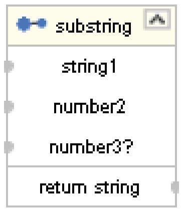
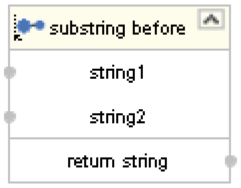
Symbol	Name	Function
	substring after	Returns the part of the string in the string argument that occurs after the substring in the substring argument
	translate	Performs a character by character replacement. It looks in the value argument for characters contained in string1, and replaces each character for the one in the same position in the string2
	concat	Returns the concatenation of all its arguments
	copy from	Allows you to type in xpath expression for the source of a copy operation. This is useful for entering xpath predicates. Note: This is for advanced users who are familiar with xpath and BPEL syntax
	normalize space	Removes leading and trailing spaces from a string
	string	Converts the value argument to a string

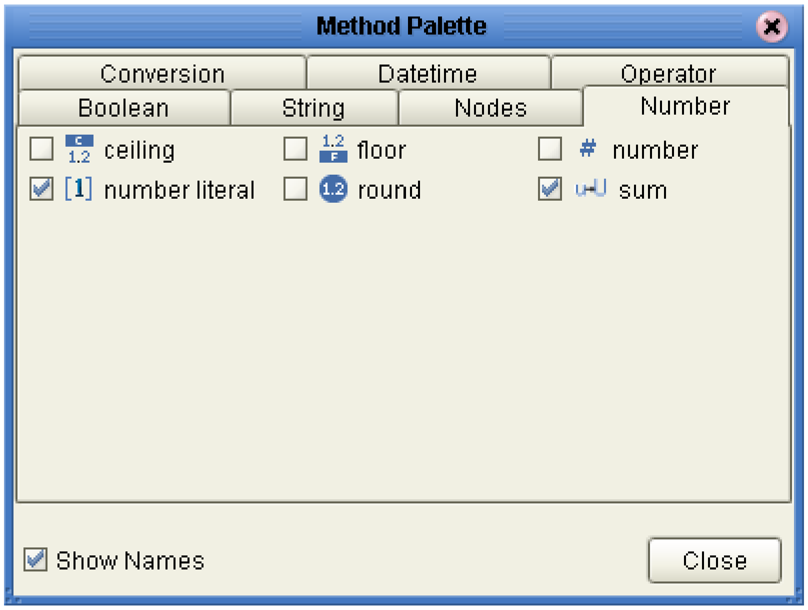
Table 43 String Methods (Continued)

Symbol	Name	Function
	string literal	A sequence of characters of fixed length and content
 substring string1 number2 number3? return string	substring	Returns a part of the string in the string argument
	substring before	Returns the part of the string in the string argument that occurs before the substring in the substring argument.

A.3 Number

The Number methods allow you to work with number data.

Figure 83 Method Palette: Number Tab



- [ ] ceiling
- [ ] floor
- [ ] number
- [x] number literal
- [ ] round
- [x] sum
- [x] Show Names

Table 44 Number Methods

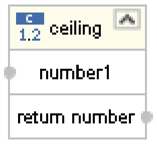
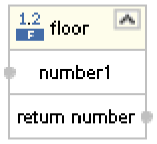
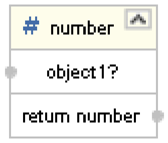
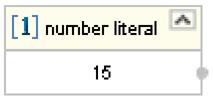
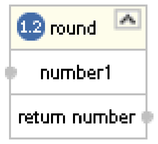
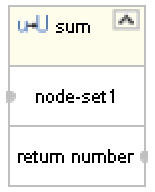
Symbol	Name	Function
	ceiling	Returns the smallest integer that is not less than the number argument
	floor	Returns the largest integer that is not greater than the number argument
	number	Converts the value argument to a number
	number literal	A literal number string of fixed length and content

Table 44 Number Methods (Continued)

Symbol	Name	Function
	round	Rounds the number argument to the nearest integer
	sum	Returns the total value of a set of numeric values in a node-set

A.4 Boolean

Boolean methods allow you to apply boolean logic to your data.

Figure 84 Method Palette: Boolean Tab

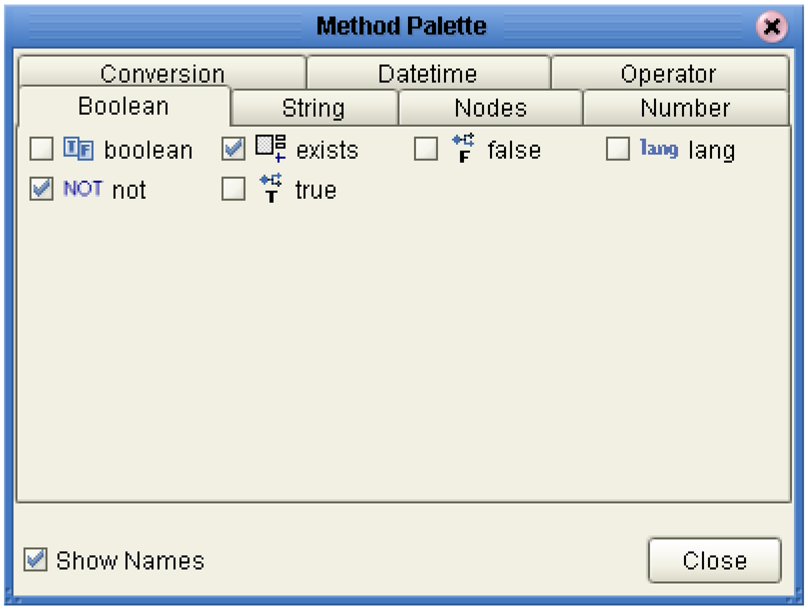
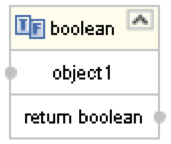
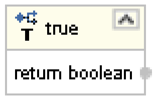
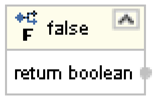
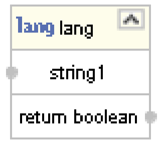
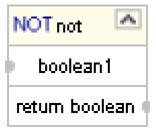
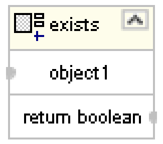


Table 45 Boolean Methods

Symbol	Name	Function
	boolean	Converts the value argument to Boolean and returns true or false.
	true	Returns true
	false	Returns false
	lang	Returns true if the language argument matches the language of the xsl:lang element, otherwise it returns false.
	not	Returns true if the condition argument is false, and false is the condition argument is true.
	exists	Checks to see if a value is present and returns a Boolean result.

A.5 Nodes

Node methods allow you to manipulate your data.

Figure 85 Method Palette: Nodes Tab

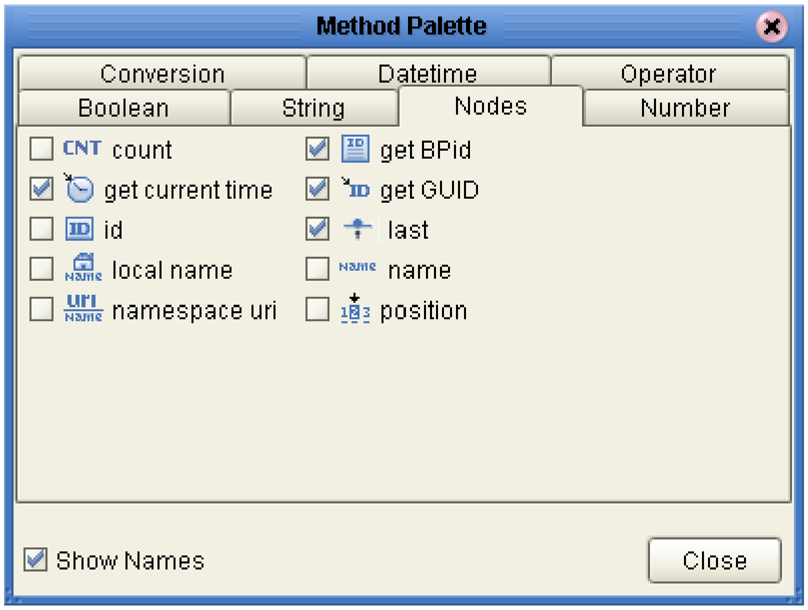


Table 46 Nodes Methods

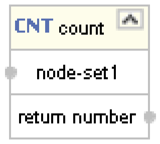
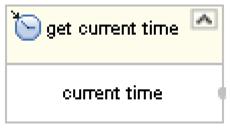
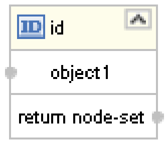
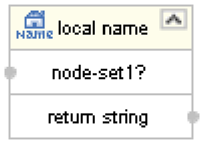
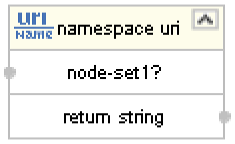
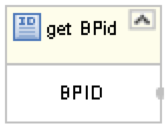
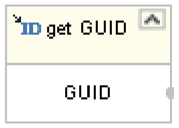
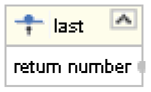
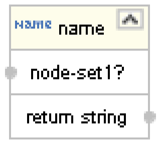
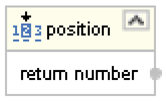
Symbol	Name	Function
	count	Returns the number of nodes in a node-set
	get current time	Gets the current time in ISO 8601 format (e.g. 2003-08-15T02:03:49.92Z).
	id	Selects elements by their unique ID
	local name	Returns the local part of a node. A node usually consists of a prefix, a colon, followed by the local name

Table 46 Nodes Methods (Continued)

Symbol	Name	Function
	namespace uri	Returns the namespace URI of a specified node
	get BPid	Gets the business process instance ID.
	get GUID	Gets a randomly generated globally unique ID.
	last	Returns the position number of the last node in the processed node list
	name	Returns the name of a node
	position	Returns the position in the node list of the node that is currently being processed

A.6 Datetime

Datetime methods allow you to manipulate date, time, and duration of data.

Figure 86 Method Palette: Datetime Tab

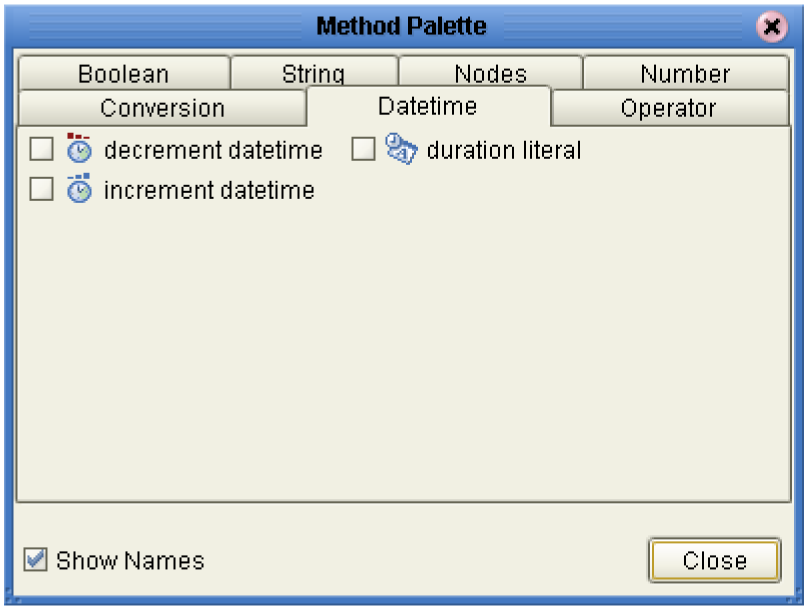
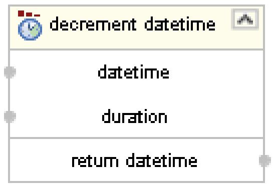
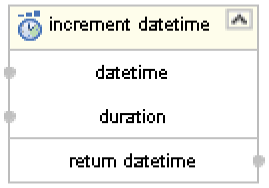
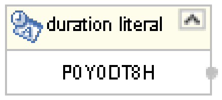


Table 47 Datetime Methods

Symbol	Name	Function
 decrement datetime datetime duration return datetime	decrement datetime	Dynamically decreases the date or time by a certain duration, such as days or hours.
 increment datetime datetime duration return datetime	increment datetime	Dynamically increases the date or time by a certain duration, such as days or hours.
	duration literal	Allows you to set an actual date or time.

A.7 Conversion

The Convert method allows you to make conversions from various data types.

Figure 87 Method Palette: Conversion Tab

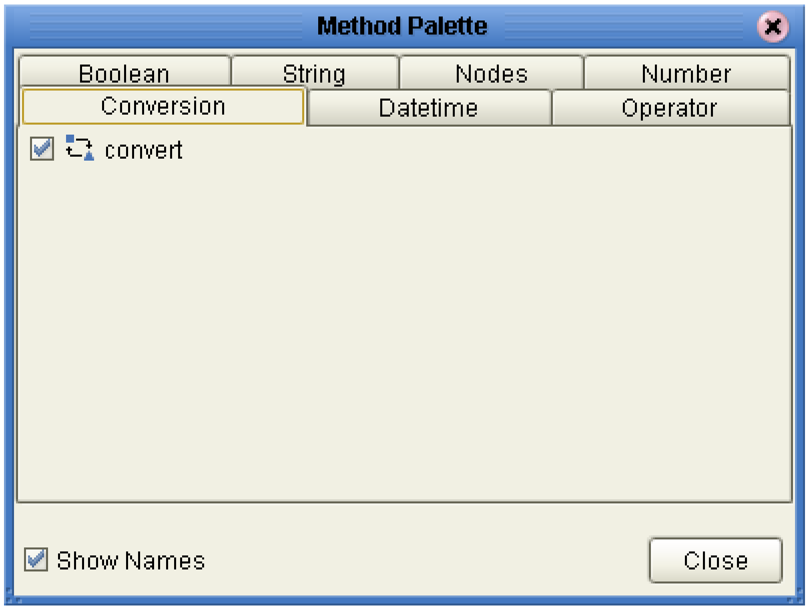
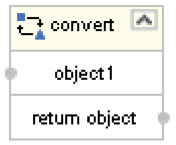


Table 48 Conversion Methods

Symbol	Name	Function
	convert	The convert function that takes in one input link and one output link. The data type conversions are described in **"Data Type Conversions" on page 157**.

A.7.1 Data Type Conversions

The eXchangeProtocol Designer supports a Convert function that takes in one input link and one output link. The Convert function is implemented from tree to tree mapping only. The Convert function is valid for conversions between leaf nodes. The Conversion function checks if the mapping is valid. The valid conversions are based off the following conversions.

String

Table 49 String

To	From
Boolean	custom
Float	parse
Double	parse
Decimal	parse
Byte	parse
Short	parse
Int	parse
Long	parse
Duration	parse
dateTime	parse
time	parse
date	parse
gYearMonth	parse
gYear	parse
gMonthDay	parse
gDay	parse
gMonth	parse
hexBinary	textToByte
base64Binary	textToByte
anyURI	parse
QName	parse
NOTATION	parse

Boolean

Table 50 Boolean

To	From
String	toString

Float

Table 51 Float

To	From
String	toString
Boolean	boolean
Double	floatToDouble
Decimal	floatToDecimal
Byte	floatToByte
Short	floatToShort
Int	floatToInt
Long	floatToLong

Double

Table 52 Double

To	From
String	toString
Boolean	boolean
Float	doubleToFloat
Decimal	doubleToDecimal
Byte	doubleToByte
Short	doubleToShort
Int	doubleToInt
Long	doubleToLong

Decimal

Table 53 Decimal

To	From
String	toString
Boolean	boolean
Float	decimalToFloat
Double	decimalToDouble
Byte	decimalToByte
Short	decimalToShort

Table 53 Decimal (Continued)

To	From
Int	decimalToInt
Long	decimalToLong

Byte

Table 54 Byte

To	From
String	toString
Boolean	boolean()
Float	byteToFloat
Double	byteToDouble
Decimal	byteToDecimal
Short	byteToShort
Int	byteToInt
Long	byteToLong

Short

Table 55 Short

To	From
String	toString
Boolean	boolean()
Float	shortToFloat
Double	shortToDouble
Decimal	shortToDecimal
Byte	shortToByte
Int	shortToInt
Long	shortToLong

Int

Table 56 Int

To	From
String	toString
Boolean	boolean()
Float	intToFloat
Double	intToDouble
Decimal	intToDecimal
Byte	intToByte
Short	intToShort
Long	intToLong

Long

Table 57 Long

To	From
String	toString
Boolean	boolean()
Float	longToFloat
Double	longToDouble
Decimal	longToDecimal
Byte	longToByte
Short	longToShort
Int	longToInt

Duration

Table 58 Duration

To	From
String	toString
Boolean	boolean

dateTime

Table 59 dateTime

To	From
String	toString
Boolean	boolean
time	dateTimeToTime
date	dateTimeToDate
gYearMonth	dateTimeToGYearMonth
gYear	dateTimeToGYear
gMonthDay	dateTimeToGMonthDay
gDay	dateTimeToGDay
gMonth	dateTimeToGMonth

time

Table 60 time

To	From
String	toString
Boolean	boolean

date

Table 61 date

To	From
String	toString
Boolean	boolean
gYearMonth	dateToGYearMonth
gYear	dateToGYear
gMonthDay	dateToGMonthDay
gDay	dateToGDay
gMonth	dateToGMonth

gYearMonth

Table 62 gYearMonth

To	From
String	toString
Boolean	boolean
gYear	gYearMonthToGYear
gMonth	gYearMonthToGMonth

gYear

Table 63 gYear

To	From
String	toString
Boolean	boolean

gMonthDay

Table 64 gMonthDay

To	From
String	toString
Boolean	boolean
gDay	gMonthDayToGDay
gMonth	gMonthDayToGMonth

gDay

Table 65 gDay

To	From
String	toString
Boolean	boolean

gMonth

Table 66 gMonth

To	From
String	toString
Boolean	boolean

hexBinary

Table 67 hexBinary

To	From
String	byteToText
Boolean	boolean
base64Binary	hexBinaryToBase64Binary

base64Binary

Table 68 base64Binary

To	From
String	byteToText
Boolean	boolean
hexBinary	base64BinaryToHexBinary

anyURI

Table 69 anyURI

To	From
String	toString
Boolean	boolean

QName

Table 70 QName

To	From
String	toString
Boolean	boolean

NOTATION

Table 71 NOTATION

To	From
String	toString
Boolean	boolean

Index