SUN SEEBEYOND

COBOL COPYBOOK CONVERTER USER'S GUIDE

Release 5.1.1

Sun microsystems

Copyright © 2006 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved. Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at http://www.sun.com/patents and one or more additional patents or pending patent applications in the U.S. and in other countries. U.S. Government Rights - Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements. Use is subject to license terms. This distribution may include materials developed by third parties. Sun, Sun Microsystems, the Sun logo, Java, Sun Java Composite Application Platform Suite, SeeBeyond, eGate, eInsight, eVision, eTL, eXchange, eView, eIndex, eBAM, eWay, and JMS are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon architecture developed by Sun Microsystems, Inc. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd. This product is covered and controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.

Copyright © 2006 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, Etats-Unis. Tous droits réservés. Sun Microsystems, Inc. détient les droits de propriété intellectuels relatifs à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et ce sans limitation, ces droits de propriété intellectuels peuvent inclure un ou plus des brevets américains listés à l'adresse http://www.sun.com/patents et un ou les brevets supplémentaires ou les applications de brevet en attente aux Etats - Unis et dans les autres pays. L'utilisation est soumise aux termes de la Licence. Cette distribution peut comprendre des composants développés par des tierces parties. Sun, Sun Microsystems, le logo Sun, Java, Sun Java Composite Application Platform Suite, Sun, SeeBeyond, eGate, eInsight, eVision, eTL, eXchange, eView, eIndex, eBAM et eWay sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd. Ce produit est couvert à la législation américaine en matière de contrôle des exportations et peut être soumis à la règlementation en vigueur dans d'autres pays dans le domaine des exportations et importations. Les utilisations, ou utilisateurs finaux, pour des armes nucléaires, des missiles, des armes biologiques et chimiques ou du nucléaire maritime, directement ou indirectement, sont strictement interdites. Les exportations ou réexportations vers les pays sous embargo américain, ou vers des entités figurant sur les listes d'exclusion d'exportation américaines, y compris, mais de manière non exhaustive, la liste de personnes qui font objet d'un ordre de ne pas participer, d'une façon directe ou indirecte, aux exportations des produits ou des services qui sont régis par la législation américaine en matière de contrôle des exportations et la liste de ressortissants spécifiquement désignés, sont rigoureusement interdites.

Version 20060614140928

Contents

Chapter 1

Introducing the COBOL Copybook Converter

Welcome to the *Sun SeeBeyond COBOL Copybook Converter User's Guide*. This document includes information about installing, configuring, and using the Sun Java Composite Application Platform Suite COBOL Copybook Converter.

This chapter provides an overview of COBOL Copybook data and data specifications. This chapter also introduces the COBOL Copybook Converter.

What's in This Chapter

1.1 About COBOL Copybooks

Copybooks are common fragments of code that are typically distributed throughout a software application. Functionally similar to the #include file of a C or C++ application, mainframes reference these books, which are usually stored in a source library file, and call structures as needed. When integrating mainframe applications with other platforms, it is necessary to retrieve and generate the data structure of the copybook. Without the copybook's data structure, your disparate applications are not able to communicate with each other and are not capable of transferring data between applications and platforms.

1.1.1 Copybooks with content beyond column 72

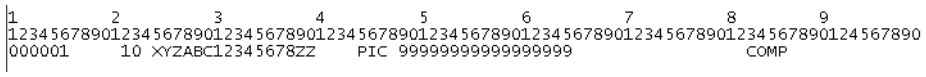
The content of the COBOL Copybook, compliant with the IBM COBOL Reference standard, does not go past column 72. To process a copybook that contains data beyond

column 72 (e.g., content that is not line numbering or comments, which should be ignored), deselect the **Ignore copybook content beyond column 72** option.

Caution: *It is still possible for a copybook with data beyond the 72th column to process successfully--but not correctly--if the latter option is selected.*

Figure 1 demonstrates copybook content beyond column 72 that may be incorrectly processed.

Figure 1 Copybook content beyond column 72.



If you disable content past column 72, the word "COMP" that begins in column 73 is ignored. Even without this word, the content that appears within the first 72 columns composes a correct (but now misinterpreted) description entry. With the option selected, the entry describes *XYZABC12345678ZZ* as a 18-character alpha-numeric item, using 18 bytes of storage (implicit USAGE is DISPLAY). With the option disabled, the entry describes a 18-digit numeric item using 8 bytes of storage (USAGE is COMP).

1.2 About the COBOL Copybook Converter

The COBOL Copybook Converter converts copybook descriptions, and creates OTDs designed to encapsulate data conforming to the description. The generated OTD is a model, containing a user-friendly abstraction of the data. COBOL Copybook Converter OTDs enable you to handle the data, which is COBOL/EBCDIC in form, as objects of the Java programming language.

The COBOL Copybook Converter reads the copybook specification from a flat file. The Converter feature uses the 01 segment of the COBOL copybook as the root node of the OTD. For example, if you are using a CICS eWay, after you have generated an OTD file, the eGate Project can populate the file and present it into the COMM AREA for CICS calls. Similarly, the system can parse the output COMM AREA from CICS into OTDs created by the COBOL Copybook Converter.

For further information on features supported by the COBOL Copybook Converter, see **Appendix A**.

Note: *The COBOL Copybook Converter must have valid COBOL syntax to complete an accurate conversion. The COBOL Copybook Converter performs limited syntax validation on an input copybook. To ensure a functional OTD conversion, verify that the copybook supplied to the Converter is well-formed* ***and*** *valid.*

1.2.1 Unsupported Features

The following COBOL Copybook features are not supported by the COBOL Copybook Converter:

- **COBOL Copy Statements** — COBOL copy statements that are embedded within the COBOL Copybook are not supported.
- **Usage Pointer** — Usage pointer statements are not supported. To accommodate these elements, you must change the statement to **PIC X(4)**. The COBOL Copybook Converter interprets this and creates a node of the correct length with the subsequent nodes as siblings instead of child nodes.
- **Complete COBOL programs** — these contain non-working storage and non-linkage areas (such as an Environment Division area). The COBOL Copybook Converter processes COBOL files with working-storage and linkage-section record entries only.

1.3 What's New in This Release

The COBOL Copybook Converter 5.1.1 includes the following changes and new features:

- Version Control: An enhanced version control system allows you to effectively manage changes to the eWay components.
- Multiple Drag-and-Drop Component Mapping from the Deployment Editor: The Deployment Editor now allows you to select multiple components from the Editor's component pane, and drop them into your Environment component.
- Relaunchable OTD Support: An OTD can be rebuilt and saved (under the same name) then relaunched back to the same Java Collaboration or BPEL. This allows you to change the metadata in an OTD without having to completely recreate the business logic from scratch.
- Connectivity Map Generator: Generates and links your Project's Connectivity Map components using a Collaboration or Business Process.

Many of these features are documented further in the *Sun SeeBeyond eGate™ Integrator User's Guide* or the *Sun SeeBeyond eGate™ Integrator System Administration Guide*.

1.4 About This Document

This document includes the following chapters:

- **Chapter 1 "Introducing the COBOL Copybook Converter":** Provides an overview description of the product as well as high-level information about this document.

- **Chapter 2 "Installing the COBOL Copybook Converter":** Describes how to install the COBOL Copybook Converter and its sample Project.
- **Chapter 3 "Using the COBOL Copybook Converter OTD Wizard"**: Describes how to use the OTD wizard to create and configure Object Type Definitions.
- **Chapter 4 "Implementing the COBOL Copybook Converter Sample Projects":** Describes how to use the COBOL Copybook Converter. The chapter also includes procedures for importing and using the COBOL Copybook sample Project.
- **Chapter A "Frequently Asked Questions"**: Provides a listing of frequently asked questions about COBOL Copybook Converter.

1.4.1 Scope

This user's guide provides a description of the COBOL Copybook Converter. It includes directions for installing the Converter, configuring the Converter properties, and implementing the Converter's sample Projects. This document is also intended as a reference guide, listing available properties, functions, and considerations.

1.4.2 Intended Audience

This guide is intended for experienced computer users who have the responsibility of helping to set up and maintain a fully functioning Java Composite Application Platform Suite system. This person must also understand any operating systems on which the Java Composite Application Platform Suite will be installed (Windows and UNIX), and must be thoroughly familiar with Windows-style GUI operations.

1.4.3 Text Conventions

The following conventions are observed throughout this document.

Table 1 Text Conventions

Text Convention	Used For	Examples
Bold	Names of buttons, files, icons, parameters, variables, methods, menus, and objects	▪ Click **OK**. ▪ On the **File** menu, click **Exit**. ▪ Select the **eGate.sar** file.
`Monospaced`	Command line arguments, code samples; variables are shown in ***bold italic***	`java -jar filename.jar`
Blue bold	Hypertext links within document	See **Text Conventions** on page 9
Blue underlined	Hypertext links for Web addresses (URLs) or email addresses	http://www.sun.com

1.5 Related Documents

The following Sun documents provide additional information about the Sun Java Composite Application Platform Suite product:

- *Sun SeeBeyond eGate™ Integrator User's Guide*
- *Sun Java Composite Application Platform Suite Installation Guide*

1.6 Sun Microsystems, Inc. Web Site

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

http://www.sun.com

1.7 Documentation Feedback

We appreciate your feedback. Please send any comments or suggestions regarding this document to:

CAPS_docsfeedback@sun.com

Chapter 2

Installing the COBOL Copybook Converter

This chapter describes how to install the COBOL Copybook Converter.

What's in This Chapter

2.1 COBOL Copybook Converter System Requirements

The COBOL Copybook Converter Readme contains the latest information on:

- Supported Operating Systems
- System Requirements
- External System Requirements

The COBOL Copybook Converter Readme is uploaded with the Converter's documentation file (**CobolCopyBookDocs.sar**) and can be accessed from the **Documentation** tab of the Sun Java Composite Application Platform Suite Installer. Refer to the COBOL Copybook Converter Readme for the latest requirements before installing the COBOL Copybook Converter.

2.2 Installing the COBOL Copybook Converter

The Sun Java Composite Application Platform Suite Installer, a web-based application, is used to select and upload add-ons and add-on files during the installation process. The following section describes how to install the components required for the Converter.

Note: *When the Repository is running on a UNIX operating system, add-ons such as the Converter are loaded from the Sun Java Composite Application Platform Suite Installer running on a Windows platform connected to the Repository server using Internet Explorer.*

2.2.1 Installing the COBOL Copybook Converter on an eGate supported system

Follow the directions for installing the Sun Java Composite Application Platform Suite in the *Sun Java Composite Application Platform Suite Installation Guide.* After you have installed eGate or eInsight, do the following:

1. From the Sun Java Composite Application Platform Suite Installer's **Select Sun Java Composite Application Platform Suite Products to Install** table (Administration tab), expand the **eWay** option.
2. Select the products for your Sun Java Composite Application Platform Suite and include the following:
   - **File eWay** (the File eWay is used by most sample Projects)
   - **COBOLCopybook**

   To upload the COBOL Copybook Converter User's Guide, Help file, Readme, and sample Projects, select the following:
   - **COBOLCopybookDocs**
3. Once you have selected all of your products, click **Next** in the top-right or bottom-right corner of the **Select Sun Java Composite Application Platform Suite Products to Install** box.
4. From the **Selecting Files to Install** box, locate and select your first product's SAR file. Once you have selected the SAR file, click **Next**. Your next selected product appears. Follow this procedure for each of your selected products. The **Installation Status** window appears and installation begins after the last SAR file has been selected.
5. Once your product's installation is finished, continue installing the Sun Java Composite Application Platform Suite as instructed in the *Sun Java Composite Application Platform Suite Installation Guide*.

Adding the eWay to an Existing Sun Java Composite Application Platform Suite Installation

If you are adding the eWay to an existing Sun Java Composite Application Platform Suite installation, do the following:

1. Complete steps 1 through 4 above.
2. Once your product's installation is complete, open the Enterprise Designer and select **Update Center** from the Tools menu. The **Update Center Wizard** appears.
3. For Step 1 of the wizard, simply click **Next**.
4. For Step 2 of the wizard, click the **Add All** button to move all installable files to the **Include in Install** field, then click **Next**.
5. For Step 3 of the wizard, wait for the modules to download, then click **Next**.
6. The wizard's Step 4 window displays the installed modules. Review the installed modules and click **Finish.**

7 When prompted, restart the IDE (Integrated Development Environment) to complete the installation.

2.2.2 Extracting the Sample Projects

The COBOL Copybook Converter includes sample Projects. The sample Projects are designed to provide you with a basic understanding of how certain database operations are performed using the Converter.

Steps to extract the Sample Projects include:

1 Click the **Documentation** tab of the Sun Java Composite Application Platform Suite Installer, then click the **Add-ons** tab.

2 Click the **COBOL Copybook Converter** link. Documentation for the COBOL Copybook Converter appears in the right pane.

3 Click the icon next to **Sample Projects** and extract the ZIP file. Note that the **COBOL_Copybook_Sample.zip** file contains two additional ZIP files for each sample Project.

Refer to **"Importing the Sample Projects" on page 34** for instructions on importing the sample Project into your repository via the Enterprise Designer.

2.3 ICAN 5.0 Project Migration Procedures

This section describes how to transfer your current ICAN 5.0.x Projects to the Sun Java Composite Application Platform Suite 5.1.1. To migrate your ICAN 5.0.x Projects to the Sun Java Composite Application Platform Suite 5.1.1, do the following:

Export the Project

1 Before you export your Projects, save your current ICAN 5.0.x Projects to your Repository.

2 From the Project Explorer, right-click your Project and select **Export** from the shortcut menu. The Export Manager appears.

3 Select the Project that you want to export in the left pane of the Export Manager and move it to the Selected Projects field by clicking the **Add to Select Items** (arrow) button, or click **All** to include all of your Projects.

4 In the same manner, select the Environment that you want to export in the left pane of the Export Manager and move it to the Selected Environments field by clicking the **Add to Select Items** (arrow) button, or click **All** to include all of your Environments.

5 Browse to select a destination for your Project ZIP file and enter a name for your Project in the **ZIP file** field.

6 Click **Export** to create the Project ZIP file in the selected destination.

Install Java CAPS 5.1.1

1. Install **Java CAPS 5.1.1**, including all eWays, libraries, and other components used by your ICAN 5.0 Projects.
2. Start the Java CAPS 5.1.1 Enterprise Designer.

Import the Project

1. From the Java CAPS 5.1.1 Enterprise Designer's Project Explorer tree, right-click the Repository and select **Import Project** from the shortcut menu. The Import Manager appears.
2. Browse to and select your exported Project file.
3. Click **Import**. A warning message, **"Missing APIs from Target Repository**," may appear at this time. This occurs because various product APIs were installed on the ICAN 5.0 Repository when the Project was created that are not installed on the Java CAPS 5.1.1 Repository. These APIs may or may not apply to your Projects. You can ignore this message if you have already installed all of the components that correspond to your Projects. Click **Continue** to resume the Project import.
4. Close the Import Manager after the Project is successfully imported.

Deploy the Project

1. A new Deployment Profile must be created for each of your imported Projects. When a Project is exported, the Project's components are automatically *"checked in"* to Version Control to write-protected each component. These protected components appear in the Explorer tree with a red padlock in the bottom-left corner of each icon. Before you can deploy the imported Project, the Project's components must first be *"checked out"* of Version Control from both the Project Explorer and the Environment Explorer. To *"check out"* all of the Project's components, do the following:
   A. From the Project Explorer, right-click the Project and select **Version Control > Check Out** from the shortcut menu. The Version Control - Check Out dialog box appears.
   B. Select **Recurse Project** to specify all components, and click **OK**.
   C. Select the Environment Explorer tab, and from the Environment Explorer, right-click the Project's Environment and select **Version Control > Check Out** from the shortcut menu.
   D. Select **Recurse Environment** to specify all components, and click **OK**.
2. If your imported Project includes File eWays, these must be reconfigured in your Environment prior to deploying the Project.

   To reconfigure your File eWays, do the following:
   A. From the Environment Explorer tree, right-click the File External System, and select **Properties** from the shortcut menu. The Properties Editor appears.
   B. Set the inbound and outbound directory values, and click **OK**. The File External System can now accommodate both inbound and outbound eWays.
3. Deploy your Projects.

Note: *Only projects developed on ICAN 5.0.2 and later can be imported and migrated successfully into the Sun Java Composite Application Platform Suite.*

Chapter 3

Using the COBOL Copybook Converter OTD Wizard

This chapter describes how to use the COBOL Copybook Converter OTD Wizard to build OTDs and introduces the Converter's OTD methods.

What's in This Chapter

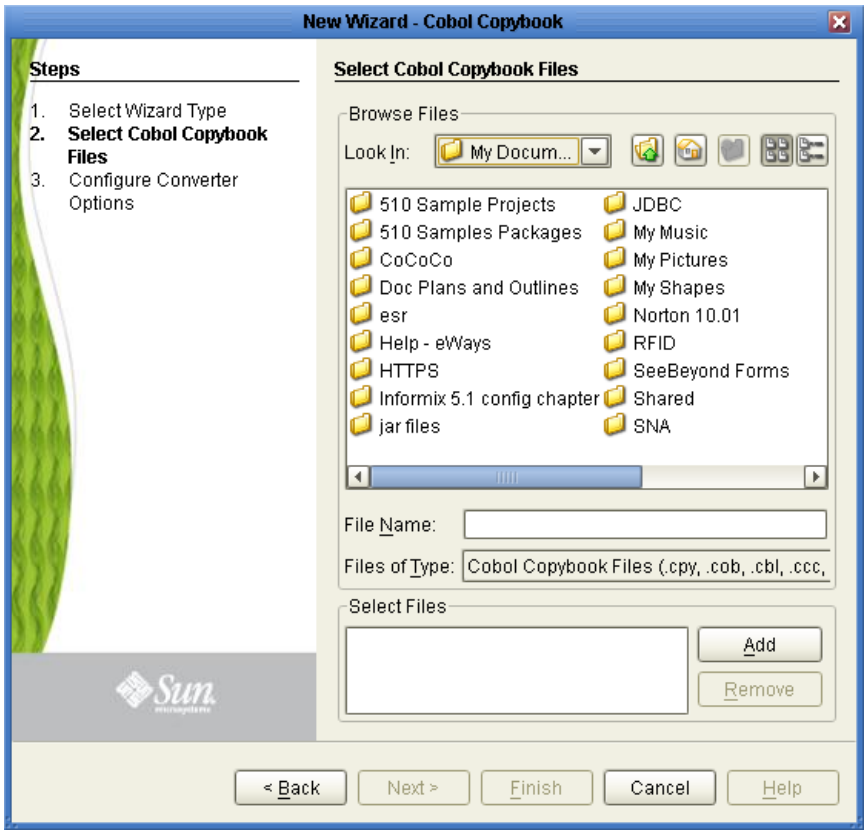
3.1 Creating COBOL Copybook OTDs

You use the COBOL Copybook wizard within Enterprise Designer to create Copybook Converter OTDs. These OTDs can then later be used in Collaboration Definitions to create the business logic behind the Collaborations.

To create COBOL Copybook OTDs

1. On the Project Explorer tree, right click the Project and select **New > Object Type Definition** from the shortcut menu. The **New Object Type Definition Wizard** window appears, displaying the available **OTD** wizards.
2. Click **COBOL Copybook** and click **Next**. The **New Wizard - Cobol Copybook** window appears.

Figure 2 COBOL Copybook Wizard—COBOL Copybook Selection



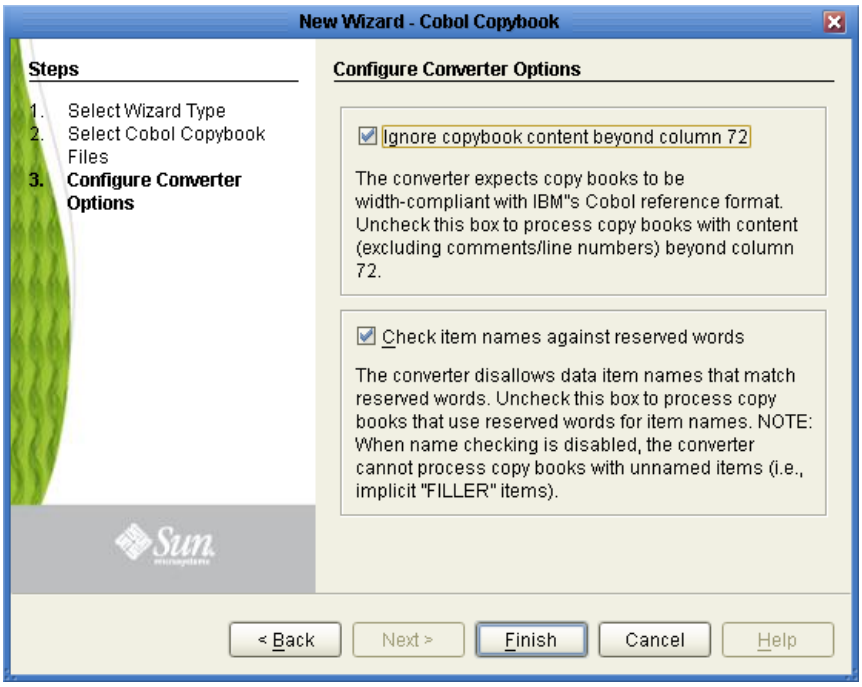
3 Browse for the desired COBOL Copybook file and highlight it.

4 Click the **Add** button to include a copybook file in a project.

5 Repeat Steps 3 and 4 for each file to include in the project. To remove a copybook file from the project, highlight the file name in the **Select Files** container and click **Remove**.

6 Click **Next**. The **Configure Converter Options** page appears.

Figure 3 COBOL Copybook Wizard—Configure Converter Options



7 Optionally, add/remove checks from boxes to enable/disable options:

- **Ignore copybook content beyond column 72** -- The Converter expects copybooks to be width-compliant with IBM's COBOL reference format. Uncheck this box to process books with content (excluding comments/line numbers) beyond column 72. Default: enabled (box is checked).
- **Check Item names against reserved words** -- The Converter disallows data item names that match reserved words. Uncheck this box to process copy books that use reserve words for item names. When name checking is disabled, the Converter cannot process copy books with unnamed items (i.e., implicit 'FILLER' items). Default: enabled (box is checked).

8 Click **Finish**. The **OTD Editor** window appears, displaying the OTD.

3.1.1 Parsing Copybook Entries

New functionality in COBOL Copybook Converter 5.1.1 allows for more accurate parsing of data entries. The Converter's parser no longer assigns globally unique names to identical data entries. If there are more than two data entries with identical names, level numbers, and same direct parent data entry, an exception will be thrown at parsing time, as follows:

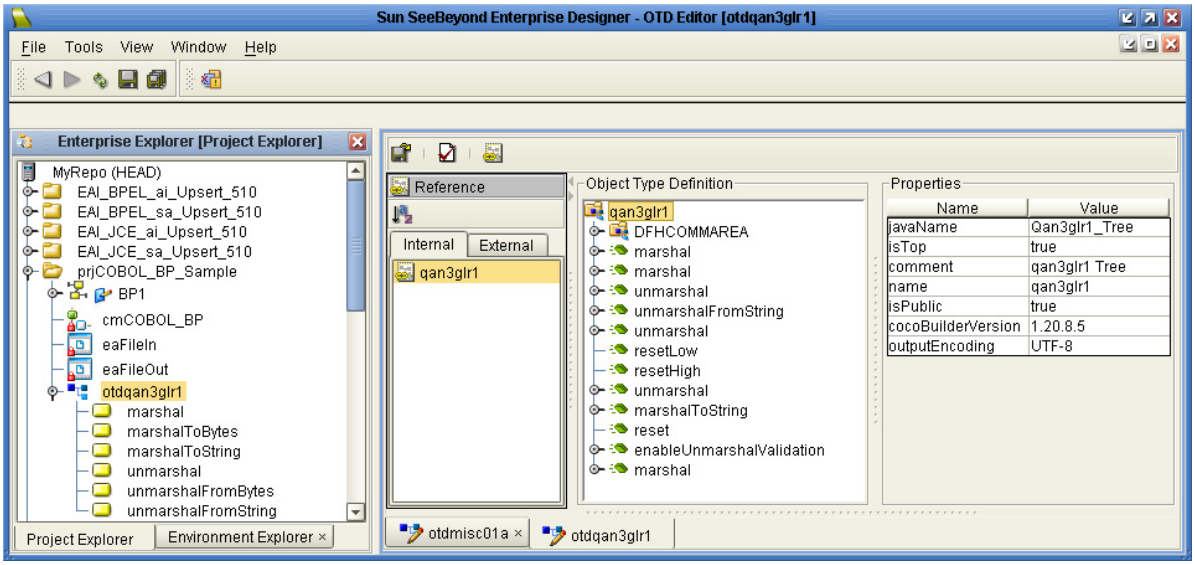
```
com.stc.cococo.builder.CocoParseException: CCCB4200:

Parse exception at line 126, column 64, item (n/a), token (n/a):

CCCB4201: Copybook item processing error.

CCCB4228: Identical data name '9 DEDUCTIBLE-LOSS-SETTLMNT-COOL FQN =
FORMATTER-COPYBOOK:SEEB-GROUP-LST-END-3-0006RG:NEW-IMPORT-GROUP-LST-
```

```
END-3:IMPORT-GRP-LST-END-3-0026EV:IE01-ENDORSEMENT-SUBJEC-
0026ET:DEDUCTIBLE-LOSS-SETTLMNT-COOL' found under parent data item '7
IE01-ENDORSEMENT-SUBJEC-0026ET FQN = FORMATTER-COPYBOOK:SEEB-GROUP-
LST-END-3-0006RG:NEW-IMPORT-GROUP-LST-END-3:IMPORT-GRP-LST-END-3-
0026EV:IE01-ENDORSEMENT-SUBJEC-0026ET'
```

3.2 COBOL Copybook OTD

When an OTD is built from a copybook file, it creates an OTD which contains methods that may be used with the converted contents of the copybook business object.

Figure 4 Sample Copybook OTD



The figure above shows the Copybook Converter OTD built from the sample copybook **qan3glr1.cobol**. The OTD has a node for each of the business processes that may be performed on the converted copybook. The unmarshal method allows business processes to flow data into the copybook OTDs and access contents field-by-field.

3.3 Relaunching OTDs

A single OTD can consist of many lines of metadata. When a change to the metadata occurs in an OTD, it does not have to be recreated from scratch. Using the **Relaunch** function allows the OTD to be rebuilt and saved under the same name, then relaunched back to the same Java Collaboration Definition (JCD) or Business Process Execution Language (BPEL).

To Relaunch an Existing OTD

1 In the Enterprise Explorer, right-click on the OTD. From the submenu, click **Relaunch**. The Select Files Wizard opens.

2 Enter the File Name (or Browse and Select) that you wish to be relaunched and click **Next**.

Note: *The File Name must be identical to the original since the name is used to generate the OTD name.*

3 Continue with the Wizard as described when creating the OTD.

4 Click the **Finish** button to save the changes.

When relaunching an OTD, an existing collaboration will not be affected if:

- New columns are added
- Deleted columns are not used in the original collaboration

Note: *Validation will fail if existing collaborations are not modified when columns are renamed or deleted.*

3.4 COBOL Copybook OTD Methods

The Object Type Definitions (OTDs) created by the COBOL Copybook Converter provide the method that you can use to extract or insert content into OTDs.

This section describes the COBOL copybook methods (operations) that are available for you to use in the source code for the Collaborations or Business Activities.

3.4.1 OTD Method Guidelines

This section addresses the concerns of global behavior, effects, and assumptions inherent to most methods.

Encoding Behavior for Redefinitions

The unmarshal and marshal methods of a COBOL Copybook OTD (with the exception of the marshalToString and unmarshalFromString) have been reimplemented to heed the OTD structure's data type information. When data flows into or out of the OTD, character set encoding is applied only to the portions of the data that fall on or draw from OTD fields corresponding to items in the Copybook specification that store character data (i.e., usage display items, whether implicitly or explicitly specified). Data for other types of OTD fields are not subject to charset encoding, since these fields are capable of containing binary (non-character) data.

An ambiguity arises when an OTD field, corresponding to a usage display item, is also the object of redefinition(s) in the Copybook. Redefined items may have alternate, multiple storage types, and to deal with such an item, the OTD must decide which one of the multiple definition is in effect at the time of unmarshaling or marshaling, in relation to the available data. The current implementation of COBOL Copybook OTDs resolve this ambiguity by ignoring redefinitions. The decision whether or not to apply encoding to a field is based solely on the item's original storage specification in the Copybook.

DBCS Items

COBOL Copybook OTDs do not support any particular Double Byte Character Set (DBCS) encoding. When inserted into DBCS nodes, it will not perform inspections of data to determine what specific DBCS encoding is used by character codes or byte sequences (e.g., discerning between a double-byte and a multi-byte encoding). As a consequence:

- DBCS items are represented in the OTD by Java byte array nodes, and their content will be treated as binary "blobs" with the following rules:
  - If content is set directly to a DBCS node, it is stored as-is.
  - If the content is retrieved directly from the DBCS node, the content that was originally set is also returned as-is.
  - If content is unmarshaled via the OTD root, the portion corresponding to the DBCS node is stored as-is. It should be noted however, that correctness of the aggregate input is the responsibility of the root-level unmarshal call (e.g., do not use unmarshalFromString if the OTD contains DBCS items).
  - If the OTD's content is marshaled, the portion corresponding to the DBCS node is yielded as-is, and is excluded from any character set transcoding that character data nodes of the OTD may be subjected to.
- Copybook OTDs will not auto-truncate DBCS data. Since the OTD cannot know the specific DBCS encoding of the data, it cannot correctly truncate it at the correct character boundaries. If the content which is set directly to a DBCS node exceeds the item's width, the OTD will raise an exception.

3.4.2 Root-level Methods

The following methods are the root-level methods provided:

enableUnmarshalValidation(boolean enable) Method

The **enableUnmarshalValidation(boolean enable)** method causes the OTD to validate data flow during an unmarshal call.

Table 2 enableUnmarshalValidation(boolean enable) Method

Syntax	Throws	Examples

void enableUnmarshalValidation(boolean enable)	None.	// enable validation during unmarshal
		// call to unmarshal may raise an exception if content is not compatible
		byte[] content = ...
		OTD_1.enableUnmarshalValidation(true);
		OTD_1.unmarshal(content);
		
		// disable validation during unmarshal
		// call to unmarshal will not raise data-related exceptions
		// instead, data-related exceptions may/will occur when
		// accessing specific nodes with invalid data.
		byte[] content = ...
		OTD_1.enableUnmarshalValidation(false);
		OTD_1.unmarshal(content)

marshal() Method

The **marshal()** method serializes the OTD's content as an array of bytes. The content is encoded with the OTD's current encoding, which is the encoding specified when data was last unmarshaled (see setEncoding() and unmarshal() for additional details). If no data was unmarshaled prior to a marshal call, then the OTD defaults to EBCDIC CP037 encoding. If the OTD content is incompatible with the current encoding (this can happen when data was unmarshaled with a different encoding than the current one), a com.stc.otd.runtime.MarshalException occurs.

Table 3 marshal() Method

Syntax	Throws	Examples

byte [] marshal()	MarshalException, IOException, UnsupportedEncodingException	// populate OTD and marshal entire content in EBCDIC
		OTD_1.setField1(...
		OTD_1.setField2(...
		...
		byte[] output = OTD_1.marshal();
		
		// write ASCII data to OTD
		// edit some fields
		// marshal OTD data (still ASCII)
		byte[] content = ...
		OTD_1.unmarshal(content, "US-ASCII");
		OTD_1.setField9(...
		OTD_1.setField10(...
		byte[] output = OTD_1.marshal();
		
		// write ASCII data to OTD
		// edit some fields
		// marshal OTD data using different encoding (may fail depending on data)
		byte[] content = ...
		OTD_1.unmarshal(content, "US-ASCII");
		OTD_1.setField9(...
		OTD_1.setField10(...
		OTD_1.useEncoding("CP277");
		byte[] output = OTD_1.marshal();

marshal(String charset) Method

The **marshal(String charset)** method serializes the content of the OTD as an array of bytes. The content is encoded using the user-specified character set. The encoding specified in this call acts as a temporary override to the OTD's current encoding, but does not become the current encoding (see setEncoding and unmarshal documentation for information). If the OTD content is not compatible with the current encoding (this can happen if data was unmarshaled using an encoding different from the current one), com.stc.otd.MarshalException occurs. If the specified *charset* value does not name a supported character set, a java.io.UnsupportedEncodingException is generated.

Table 4 marshal(String charset) Method

Syntax	Throws	Examples

byte[] marshal(String charset)	MarshalException, IOException, UnsupportedEncodingException	byte[] content = cocoOtd.marshal("cp037"); // retrieve OTD content as EBCDIC data
		byte[] content = cocoOtd.marshal("US-ASCII"); // retrieve OTD content as ASCII data

marshal(OtdOutputStream out) Method

The **marshal(OtdOutputStream out)** method serializes the content of the OTD and writes it to the supplied output stream object. The output is encoded using the same user-specified encoding used when the data was last unmarshaled (see setEncoding and unmarshal documentation for additional details). If no data was unmarshaled prior to the call to marshal, then EBCDIC CP037 encoding is used. If the OTD content is not compatible with the current encoding (this can happen if the data was unmarshaled using an encoding different from the current one), com.stc.otd.MarshalException occurs. A java.io.IOException is generated if an output error occurs in attempting to write data to the stream object.

Table 5 marshal(OtdOutputStream out) Method

Syntax	Throws	Examples
void marshal(OtdOutputStream out)	MarshalException, IOException, UnsupportedEncodingException	

marshal(OtdOutputStream out, String charset) Method

The **marshal(OtdOutputStream out, String charset)** method flows data out from the OTD to the supplied stream object, using the specified charset encoding. The given encoding acts as a temporary override to the OTD's current encoding, it does not become the current encoding (see setEncoding and unmarshal documentation for information).

If the specified charset is not compatible with the OTD content (this can happen when the data was unmarshaled to the OTD using a different encoding), com.stc.otd.runtime.MarshalException occurs. If the encoding is not supported or recognized, java.io.UnsupportedEncodingException is generated.

Table 6 marshal(OtdOutputStream out, String charset) Method

Syntax	Throws	Examples
void marshal(OtdOutputStream stream, String charset)	MarshalException, IOException, UnsupportedEncodingException	

marshalToString() Method

The **marshalToString()** method serializes the content of the OTD to a String object. The String is created by decoding the byte data with the OTD's current encoding, which is the encoding specified when data was last unmarshaled (see setEncoding and unmarshal documentation for additional details). If no data was unmarshaled prior to a marshal call, then the OTD defaults to EBCDIC CP037 encoding. Only use this method with copybook OTDs built from copybooks comprised solely of usage display entries. Using this method on OTDs designed to hold binary data (e.g., packed decimal, internal decimal) may invalidate the data, because portions of the binary content may not have a suitable mapping to UTF-8. A java.io.UnsupportedEncodingException may occur if the current encoding (i.e., the encoding used by the last unmarshal call) is not capable of encoding the data. This is possible because certain charset encodings in Java are not two-way encodings (encodings that can decode or encode, but not both).

Table 7 marshalToString Method

Syntax	Throws	Examples
String marshalToString()	MarshalException, IOException, UnsupportedEncodingException	

reset() Method

The **reset()** method initializes the storage space of the OTD as follows:

- alphanumeric fields (PIC X) - blank spaces (EBCDIC value 0x40)
- numeric fields (PIC 9) - binary zero
- packed decimal fields - signed-trailing packed binary zero

Table 8 reset() Method

Syntax	Throws	Examples
void reset()	None.	

resetHigh() Method

The **resetHigh()** method initializes the entire storage space of the OTD to high bit values; each byte is initialized to 0xFF.

Table 9 resetHigh() Method

Syntax	Throws	Examples
void resetHigh()	None.	

resetLow() Method

The **resetLow()** method initializes the OTD storage space to low bit values; each byte is initialized to 0x0.

Table 10 resetLow() Method

Syntax	Throws	Examples
void resetLow()	None.	

retrieveEncoding() Method

The **retrieveEncoding()** method returns the canonical name of the current OTD encoding. The default current OTD encoding is "CP037" until it is changed by a successful useEncoding call, or by a call to one of the encoding-specifiable unmarshal methods. The canonical name may differ from the one used previously to set the current encoding. See the Java 2 API documentation for java.nio.charset. Charset for more information.

Table 11 retrieveEncoding() Method

Syntax	Throws	Examples
String retrieveEncoding()	None.	

unmarshal(byte[] in) Method

The **unmarshal(byte[] in)** method deserializes the given input into an internal data tree. Data flowed to the OTD using this method must use EBCDIC CP037 encoding. This method sets the OTD's current encoding to EBCDIC CP037, which is used when data is subsequently marshaled without an overriding encoding; e.g., as allowed in a marshal(OtdOutputStream, String) call.

Table 12 unmarshal(byte[] in) Method

Syntax	Throws	Examples
void unmarshal(byte[] in)	UnmarshalException, IOException	

unmarshal(OtdInputStream in) Method

Th **unmarshal(OtdInputStream in)** method populates the OTD using the supplied OtdInputStream object as the data source. The supplied object must be an opened stream with available data. A com.stc.otd.runtime.UnmarshalException is generated if the data obtained from the stream is incompatible with the OTD, and a java.io.IOException is generated if any other input error occurs in attempting to read data from the stream object. The stream object must flow data encoded in EBCDIC

CP037. This method sets the OTD's current encoding to EBCDIC CP037, which is used when data is subsequently marshaled without overriding encoding; e.g., as allowed in a marshal (OtdOutputStream, String) call.

Table 13 unmarshal(OtdInputStream in) Method

Syntax	Throws	Examples
void unmarshal(OtdInputStream in)	UnmarshalException, IOException	

unmarshal(OtdInputStream in, String charset) Method

The **unmarshal(OtdInputStream in, String charset)** method flows data to the OTD from the supplied Stream object. The stream must be open and have available data. The charset argument specifies the encoding of the stream data. The specified encoding becomes the current encoding of the OTD and is used when data is subsequently marshaled without overriding encoding; e.g., as allowed in a marshal(OtdOutputStream, String) call.

If the stream data is incompatible with the OTD, a com.stc.otd.runtime.UnmarshalException is generated. If the stream data cannot be read, a java.io.IOException is generated. If the charset value does not name a supported charset, or if it names a supported charset with one-way encoding (capable of decoding or encoding, but not both), a java.io.UnsupportedEncodingException is generated.

Table 14 unmarshal(OtdInputStream in, String charset) Method

Syntax	Throws	Examples
void unmarshal(OtdInputStream in, String charset)	UnmarshalException, IOException, UnsupportedEncodingException	

unmarshal(byte[] in, String charset) Method

This method populates the OTD using the data supplied in the byte array in. The charset argument specifies the encoding of the given data. The specified encoding becomes the current encoding of the OTD, and is used when data is subsequently marshaled without an overriding encoding; e.g., as allowed in a marshal (OtdOutputStream, String) call. If the specified charset value does not name a supported character set or names a supported charset with one-way encoding (one that can decode or encode, but not both), a java.io.UnsupportedEncodingException is generated.

Table 15 unmarshal(byte[] in, String charset) Method

Syntax	Throws	Examples
void unmarshal(byte[] in, String charset)	Unmarshal Exception, IOException, UnsupportedEncodingException	byte[] bytes = ... cocoOtd.unmarshal(bytes, "cp037"); // Interpret bytes content as EBCDIC data cocoOtd.unmarshal(bytes, "US-ASCII"); // Interpret bytes content as ASCII data

unmarshalFromString(String in) Method

The **unmarshalFromString(String in)** method populates the OTD using the specified String object as the input source. This method is useful only to unmarshal data to copybook OTDs comprised solely of character-data records (entries specified implicitly or explicitly as USAGE DISPLAY). The current OTD encoding (see setEncoding and unmarshal document for additional details) is used to encode the String's bytes.

Table 16 unmarshalFromString(String in) Method

Syntax	Throws	Examples
void unmarshalFromString(String in)	UnmarshalException, IOException	

useEncoding(String enc) Method

Use the **useEncoding(String enc)** method to designate a particular encoding to be used as the OTD's current encoding. The current OTD encoding is used when the OTD is marshaled without an overriding encoding, which is permitted for the marshal (OtdOutputStream, String) method.

An OTD's current encoding is initially EBCDIC (CP037) when it is instantiated. There are two ways to change it:

1. Unmarshaling the data, whereby the data's stated encoding becomes the current encoding.
2. Using this method to specify it.

Changing the encoding through the use of this method causes reset() to be subsequently (and automatically) called, causing the OTD's existing content to be erased. This behaviour exists to avoid situations where data, successfully unmarshaled with one charset, fails to marshal under a different charset, due to the absence of codepoint mappings between the two encodings. Use the marshal(String) method when data, which flowed in using a charset, must then be flowed out with a different charset.

If the specified encoding is the same as the current OTD encoding, the call returns without affecting the OTD's state (i.e., reset() is not called) and the data and current encoding will remain unchanged.

If the specified encoding is not supported, or is not a two-way encoding (one that can decode or encode, but not both), a java.io.UnsupportedEncodingException is thrown.

Table 17 useEncoding(String enc) Method

Syntax	Throws	Examples
void useEncoding(String enc)	UnsupportedEncodingException	

3.4.3 Non-Root Methods

Every leaf node in a COBOL Copybook OTD represents an elementary item in the Copybook source. For every given leaf node, the OTD provides “getter” and “setter” methods of which the return type and input types depend on the data type and usage type specified in the copybook for the elementary item to which the node corresponds.

For a given non-repeating leaf node named Datum, the following method forms are provided, where *T* is determined from the follow table.

- *T* getDatum()
- void setDatum(*T*)

Table 18 Datum Method Forms

Data Types	Display	COMP or COMP-4	COMP-1	COMP-2	COMP-3	COMP-5	INDEX
Alphabetic For example: PIC AAA	String						
Alphanumeric For example: PIC X9	String		String	String			
Alphanumeric edited For example: PIC XB9	String						
Numeric edited For example: PIC ZZZ99	String						
DBCS For example PIC GGBGG	byte[]						
External floating point For example: PIC +9V99E+99	BigDecimal						

Table 18 Datum Method Forms (Continued)

Data Types	Display	COMP or COMP-4	COMP-1	COMP-2	COMP-3	COMP-5	INDEX
Numeric integer (9 digits or less)	int	int			int		int
Numeric floating point (COMP-1 or COMP-2 items)	BigDecimal						
Numeric Integer (10 to 18 digits)	long	long			long	long	
Numeric integer (19 digits or more)	BigDecimal	Big Decimal			Big Decimal	Big Decimal	

For repeating leaf nodes, these two alternative methods are provided:

- *T* getDatum(int i)
- void setDatum(int *i*, *T*)

where *i* is expected to be a value from 0, representing the ordinal of the desired repetition instance, and where *T* is determined as previously described.

3.4.4 BPEL Operations

When using eInsight to process COBOL copybooks, the operations in Table are available.

Table 19 BPEL COBOL Operations

eInsight Operation	Activity
Marshal	Allows you to marshal an OTD instance to a string.
MarshalToBytes	Marshals an OTD instance (or OTD tree) to byte array using current encoding (CP037).
MarshalToString	Marshals an OTD instance (or OTD tree) to string using current encoding (CP037).
Unmarshal	This operation is retained for purposes of compatibility with the previous release of the COBOL Copybook Converter. The **Unmarshal** operation allows you to select unmarshaling from byte array or from string.
UnmarshalFromBytes	Unmarshals data from byte array into an OTD instance.
UnmarshalFromString	Unmarshals data from string into an OTD instance.

Important: *It is recommended that you use the* ***Marshal*** *and* ***Unmarshal*** *methods since they allow for more control over the output data. Both methods are available for purposes of increased compatibility.*

Chapter 4

Implementing the COBOL Copybook Converter Sample Projects

This chapter describes how to use the COBOL Copybook Converter to convert COBOL copybooks into OTDs. It also includes how to use the samples that come with the COBOL Copybook Converter. The samples are designed to provide an overview of the basic functionality of the Converter by identifying how information passes through eGate.

It is assumed that you understand the basics of creating a Project using Enterprise Designer. For more information on creating an eGate Project, see the *eGate Tutorial* and the *eGate Integrator User's Guide.*

What's in This Chapter

4.1 About the COBOL Copybook Converter Sample Projects

The COBOL Copybook Converter utility includes two sample Projects that provide basic instruction on using COBOL Copybook Converter methods in the JCD or BPEL.

The **prjCOBOL_JCD_Sample** sample Project provides an implementation of the COBOL Copybook Converter that uses Java Collaborations to execute the desired business logic. The sample Project consists of the following:

- **prjCOBOL_JCD_Sample.zip** - The project that needs to be imported to your Enterprise Designer.
- **misc01a.cobol** - The COBOL copybook file used for the conversion to create the OTD.

- **inputcobolJCD.txt.~in** - The input file that the sample Project requires when it is deployed.
- **COBOL_JCDoutput1.dat** - The expected output when the Project is executed with the given input file.

This sample Project converts EBCDIC input data to the format specified in the copybook. The input data is provided by a File eWay. This data is read into a COBOL Copybook OTD generated from the same copybook. The Collaboration shows the use of the COBOL Copybook OTD to retrieve the EBCDIC data as Java Strings for concatenation and forwards the output to an outbound File eWay. The resulting file output is the ASCII translation of the original input data.

The **prjCOBOL_BP_Sample** sample Project provides an implementation of the COBOL Copybook Converter that uses BPEL functionality. The sample Project consists of the following:

- **prjCOBOL_BP_Sample** - The project that needs to be imported to your Enterprise Designer.
- **qan3glr1.cobol** - The COBOL copybook file used for the conversion to create the OTD.
- **INPUTCOBOLbp.txt.~in** - The input file that the sample project requires when it is run.
- **COBOLBPoutput0.dat** - The expected output when the Project is executed with the given input file.

Each Project contains the following:

- Input data
- Connectivity Maps
- Collaborations
- Business Processes

Version Support

Consult the COBOL Copybook Converter **Readme** provided in the Converter's SAR file for specific ESR requirements (if they exist) to import or run each of the sample Projects.

4.1.1 Assigning Operations in JCD

The Converter's operations are listed as methods in the JCD. Perform the following steps to access these methods:

1. Create a Collaboration that contains an OTD.
2. Right-click the OTD listed in your Collaboration and then select **Select Method to Call** from the shortcut menu.
3. Browse to and select a method to call.

4.1.2 Assigning Operations in BPEL

You can associate an eInsight Business Process Activity with the Converter, both during the system design phase and during runtime. To make this association:

1. Select the desired **receive** or **write** operation under the eWay in the Enterprise Explorer.
2. Drag the operation onto the eInsight Business Process canvas.

   The operation automatically changes to an Activity with an icon identifying the component that is the basis for the Activity.

At run time, the eInsight engine invokes each step in the order that you defined in the Business Process. Using the engine's Web Services interface, the Activity in turn invokes the eWay.

4.1.3 About the eInsight Engine and eGate Components

You can deploy an eGate component as an Activity in an eInsight Business Process. Once you have associated the desired component with an Activity, the eInsight engine can invoke it using a Web Services interface.

Examples of eGate components that can interface with eInsight in this way are:

- Object Type Definitions (OTDs)
- An eWay
- Collaborations

Using the eGate Enterprise Designer and eInsight, you can add an Activity to a Business Process, then associate that Activity with an eGate component, for example, an eWay. When eInsight run the Business Process, it automatically invokes that component via its Web Services interface.

4.2 Importing the Sample Projects

The sample Projects are included as part of the installation CD-ROM package. To import a sample Project to the Enterprise Designer, do the following:

1. Extract the samples from the Sun Java Composite Application Platform Suite Installer to a local file.

   Sample files are uploaded with the eWay's documentation SAR file, and then downloaded from the Installer's **Documentation** tab. The **COBOL_Copybook_Sample.zip** file contains the various sample Project ZIP files.

Note: *Make sure you save all unsaved work before importing a Project.*

2. From the Enterprise Designer's Project Explorer pane, right-click the Repository and select **Import Project** from teh shortcut menu. The **Import Manager** appears.

3 Browse to the directory that contains the sample Project ZIP file. Select the sample file and click **Import**.

4 Click **Close** after successfully importing the sample Project.

4.3 Running the Sample Projects

The following steps are required to run the sample Projects that are contained in the **COBOLCopybookDocs.sar** file.

1 From your input directory, paste (or rename) the sample input file to triger the Converter.

2 Import the sample Projects.

3 Build, deploy, and run the sample Projects.

You must do the following before you can run an imported sample Project:

- Create an Environment
- Configure the eWays
- Create a Deployment Profile
- Create and start a domain
- Deploy the Project

4 Check the output.

4.4 Building, Deploying, and Running the prjCOBOL_JCD_Sample Project

This section provides step-by-step instructions for manually creating the prjCOBOL_JCD_Sample sample Project.

Steps required to create the sample project

- **Creating a Project** on page 36
- **Creating the OTDs** on page 36
- **Creating a Connectivity Map** on page 38
- **Creating the Collaboration Definitions (Java)** on page 39
- **Creating the Collaboration Business Rules** on page 41
- **Binding the eWay Components** on page 46
- **Creating an Environment** on page 47
- **Configuring the eWays** on page 48

4.4.1 Creating a Project

The first step is to create a new Project in the Enterprise Designer.

1. Start the Enterprise Designer.
2. From the Project Explorer tree, right-click the Repository and select **New Project**. A new Project (**Project1**) appears on the Project Explorer tree.
3. Click twice on **Project1** and rename the Project (for this sample, **prjCOBOL_JCD_Sample**).

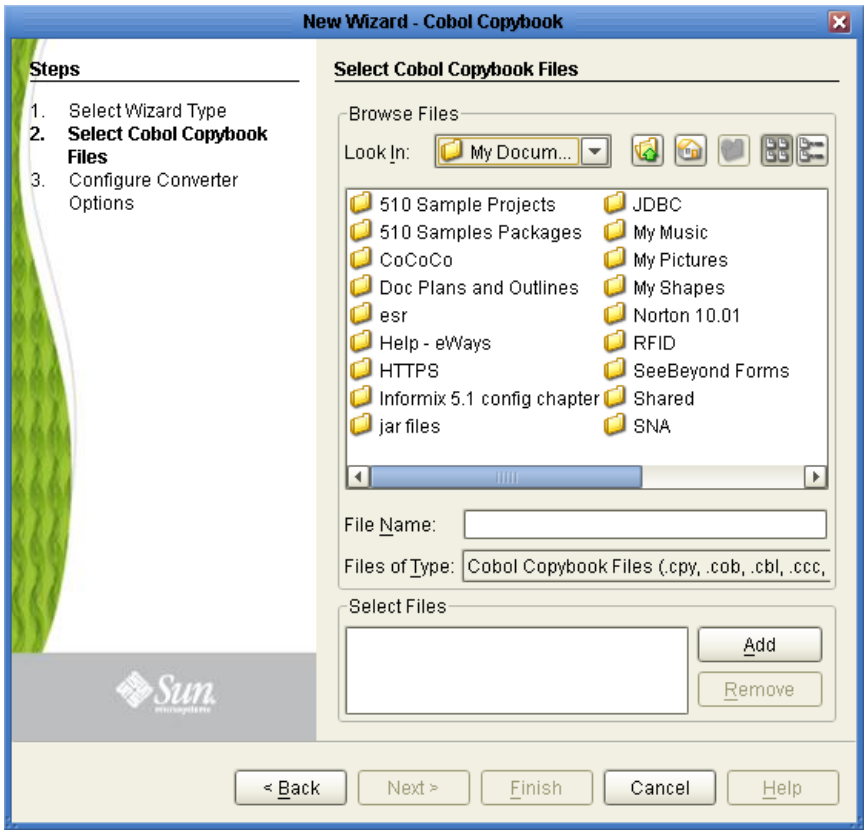
4.4.2 Creating the OTDs

The sample Project requires an OTD to interact with the COBOL Copybook Converter. The OTD for the BPEL sample Project is built on the sample COBOL file **misc01a.cobol**.

Steps required to create the COBOL misc01a OTD:

1. Right-click your new Project in the Enterprise Designer's Project Explorer, and select **New > Object Type Definition**.
2. Select **COBOL Copybook** from the list of OTD Wizards and click **Next**. The **Select COBOL Copybook Files** page appears.

Figure 5 COBOL Copybook Wizard—COBOL Copybook Selection



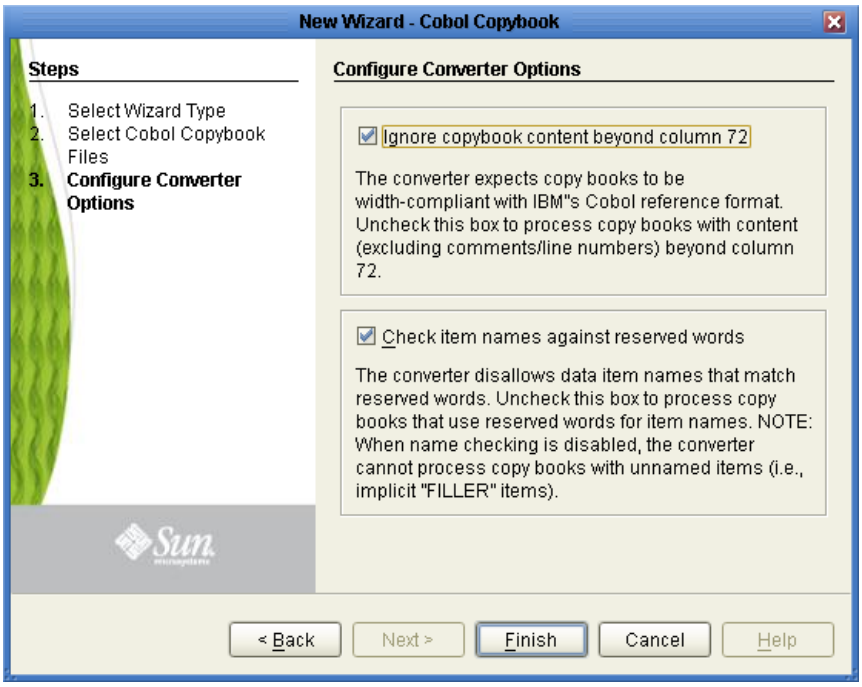
3 Browse for the desired COBOL Copybook file and highlight it.

4 Click the **Add** button to include a copybook file in a project.

5 Repeat Steps 3 and 4 for each file to include in the project. To remove a copybook file from the project, highlight the file name in the **Select Files** container and click **Remove**.

6 Click **Next**. The **Configure Converter Options** page appears.

Figure 6 COBOL Copybook Wizard—Configure Converter Options



7 Optionally, add/remove checks from boxes to enable/disable options:

- **Ignore copybook content beyond column 72** -- The Converter expects copybooks to be width-compliant with IBM's COBOL reference format. Uncheck this box to process books with content (excluding comments/line numbers) beyond column 72. Default: enabled (box is checked).
- **Check Item names against reserved words** -- The Converter disallows data item names that match reserved words. Uncheck this box to process copy books that use reserve words for item names. When name checking is disabled, the Converter cannot process copy books with unnamed items (i.e., implicit 'FILLER' items). Default: enabled (box is checked).

8 Click **Finish**. The **OTD Editor** window appears, displaying the OTD.

4.4.3 Creating a Connectivity Map

The Connectivity Map provides a canvas for assembling and configuring a Project's components.

Steps required to create a new Connectivity Map:

1 From the Project Explorer tree, right-click the new **prjCOBOL_JCD_Sample** Project and select **New > Connectivity Map** from the shortcut menu.

2 The New Connectivity Map appears and a node for the Connectivity Map is added under the Project, on the Project Explorer tree labeled **CMap1**.

The icons in the toolbar represent the available components used to populate the Connectivity Map canvas.

Populating the Connectivity Map

Add the Project components to the Connectivity Map by dragging the icons from the toolbar to the canvas.

Each Connectivity Map in the **prjCOBOL_JCD_Sample** sample Project requires the following components:

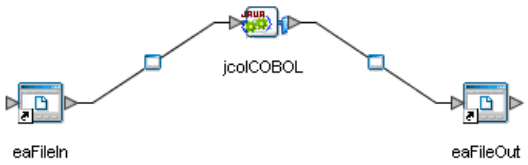
- File External Application (2)
- Service

Any eWay added to the Connectivity Map is associated with an External System.

Steps required to select a File External System:

1. Click the **External Application** icon on the Connectivity Map toolbar.
2. Select the external systems necessary to create your Project (for this sample, **File**). Icons representing the selected external system are added to the Connectivity Map toolbar.
3. Rename the following components and then save changes to the Repository:
   - File1 to eaFileIN
   - File2 to eaFileOUT

Figure 7 COBOL Copybook JCD Sample Connectivity Map



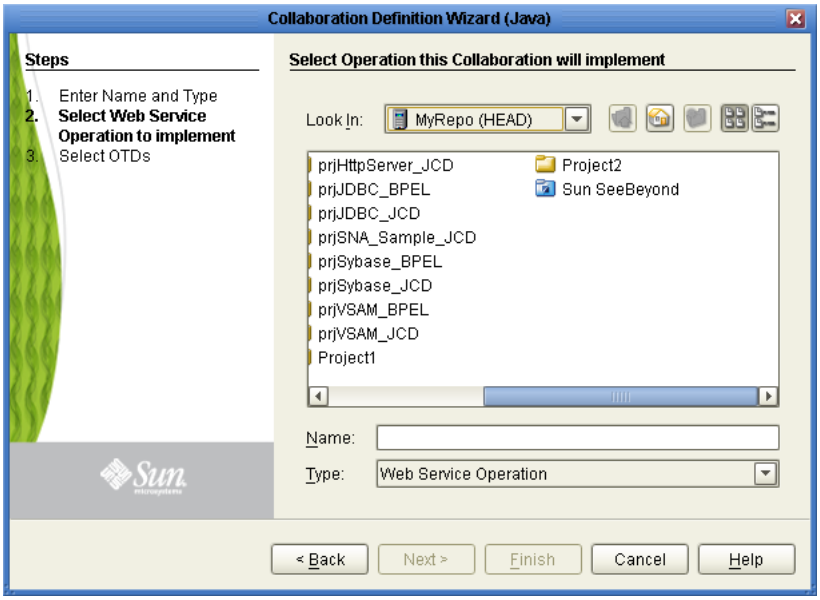
4.4.4 Creating the Collaboration Definitions (Java)

The next step is to create Collaborations using the **Collaboration Definition Wizard (Java)**. You are required to create one JCD. Once you create the Collaboration Definition, you can write the Business Rules of the Collaboration using the Collaboration Editor.

To create the Collaboration Definition

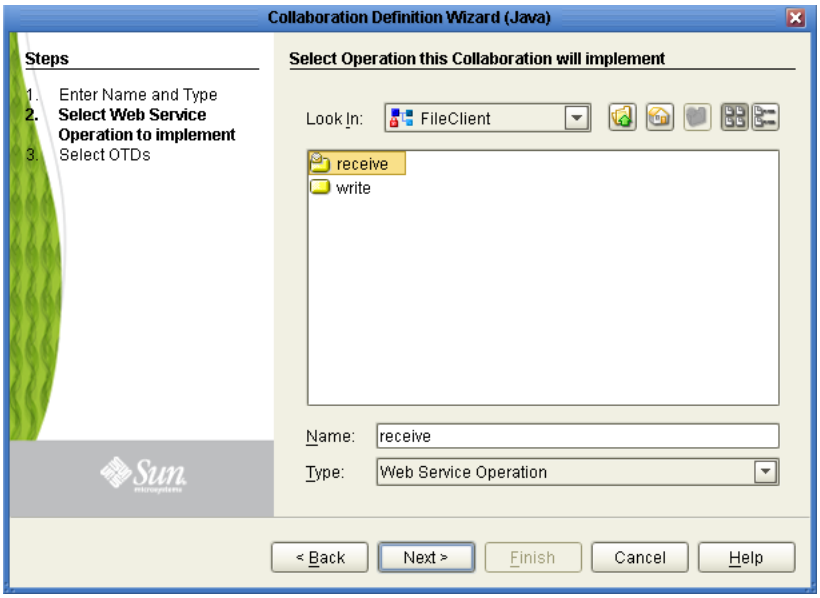
1. In the Project Explorer of the Enterprise Designer, right-click the COBOL copybook conversion Project, click **New** and click **Collaboration Definition (Java)**. The **Collaboration Definition** wizard appears.
2. In the **Collaboration Name** box, enter the name for the Collaboration and click **Next**. The **Select Operation** page appears as shown below.

Figure 8 Selecting Collaboration Operations



3 Double-click **Sun SeeBeyond** and **eWays—**continue to double-click to select the inbound eWay and the (inbound) web service. For example, for the a File eWay, double-click **File**, **FileClient**, and click **receive** as shown below.

Figure 9 Selecting File Receive



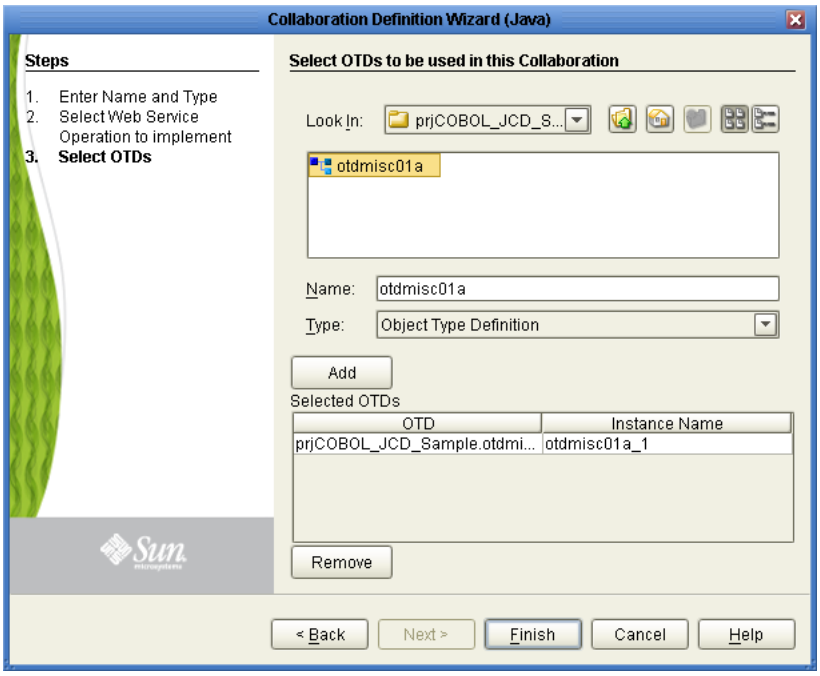
4 Click **Next**.

5 Double-click **Sun SeeBeyond**, **eWays—**continue to double-click to select the outbound eWay and the (outbound) web service. For example, for the File eWay, double-click **File**, and then **FileClient**.

6 In the **Look In** box, browse to the Project with the copybook file to be used for this conversion.

7 Double-click the copybook file. This adds the copybook file as shown below.

Figure 10 Completed Collaboration Definition



8 Click **Finish**. The **Collaboration Editor** window appears.

You can now create the business logic for the Collaboration as described below.

4.4.5 Creating the Collaboration Business Rules

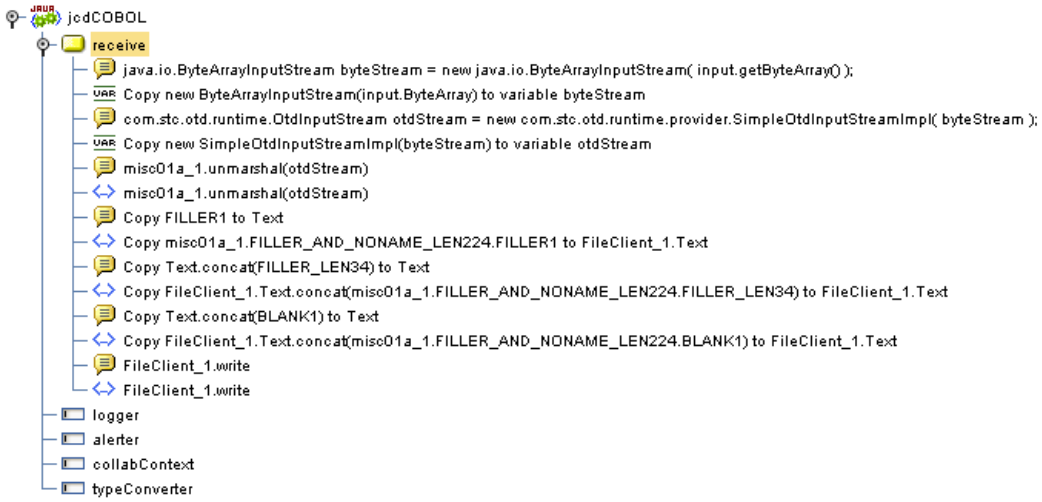
The next step in the sample is to create the Business Rules of the Collaboration using the Collaboration Editor.

Creating the prjCOBOL_JCD_Sample Business Rules

The **prjCOBOL_JCD_Sample** Collaboration implements the Input Web Service Operation to read the **inputcobolJCD.txt.~in** file and then process a COBOL Copybook. The Collaboration also writes a message to **COBOL_JCDoutput1.dat** to confirm the operation.

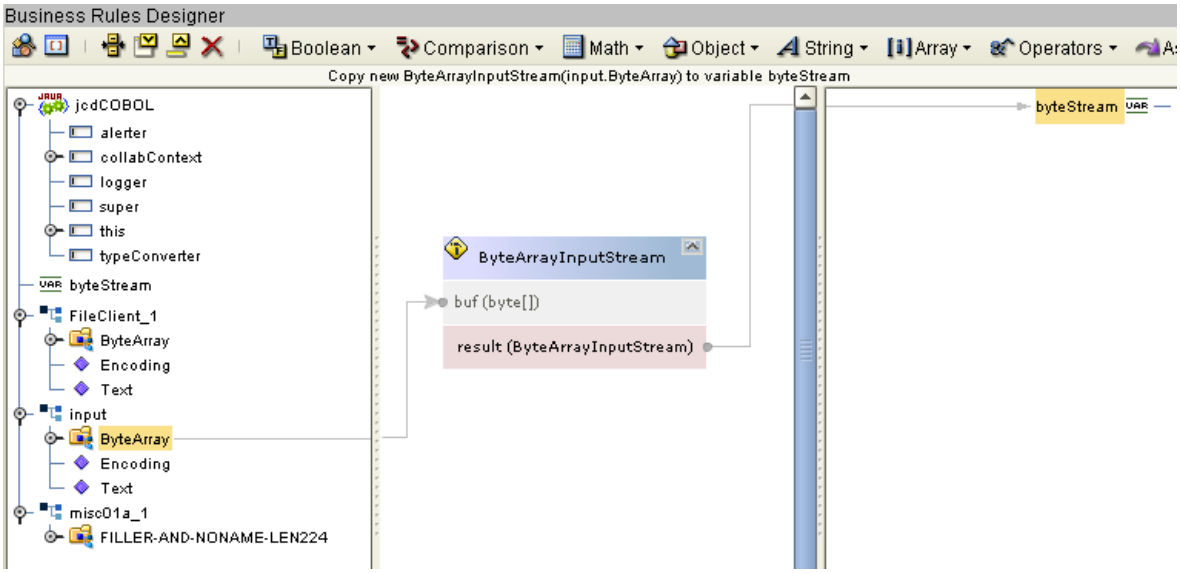
The **prjCOBOL_JCD_Sample** Collaboration contains the Business Rules displayed in Figure 11.

Figure 11 prjCOBOL_JCD_Sample Business Rules



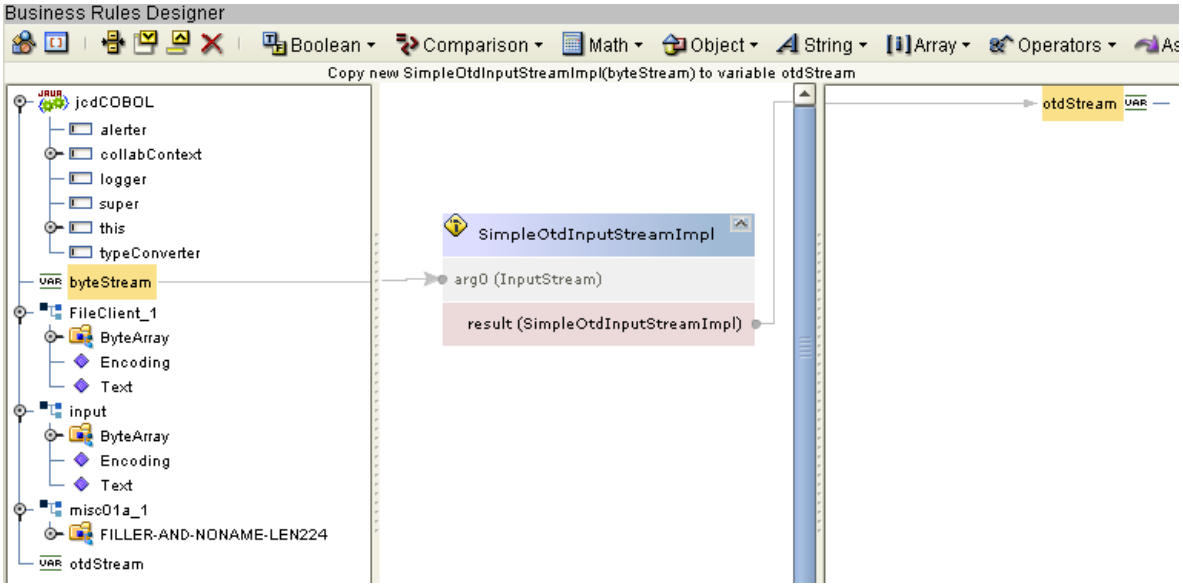
The **Copy new ByteArrayInputStream(input.ByteArray) to variable byteStream** Variable is displayed in Figure 12.

Figure 12 prjCOBOL_JCD_Sample Business Rule 1



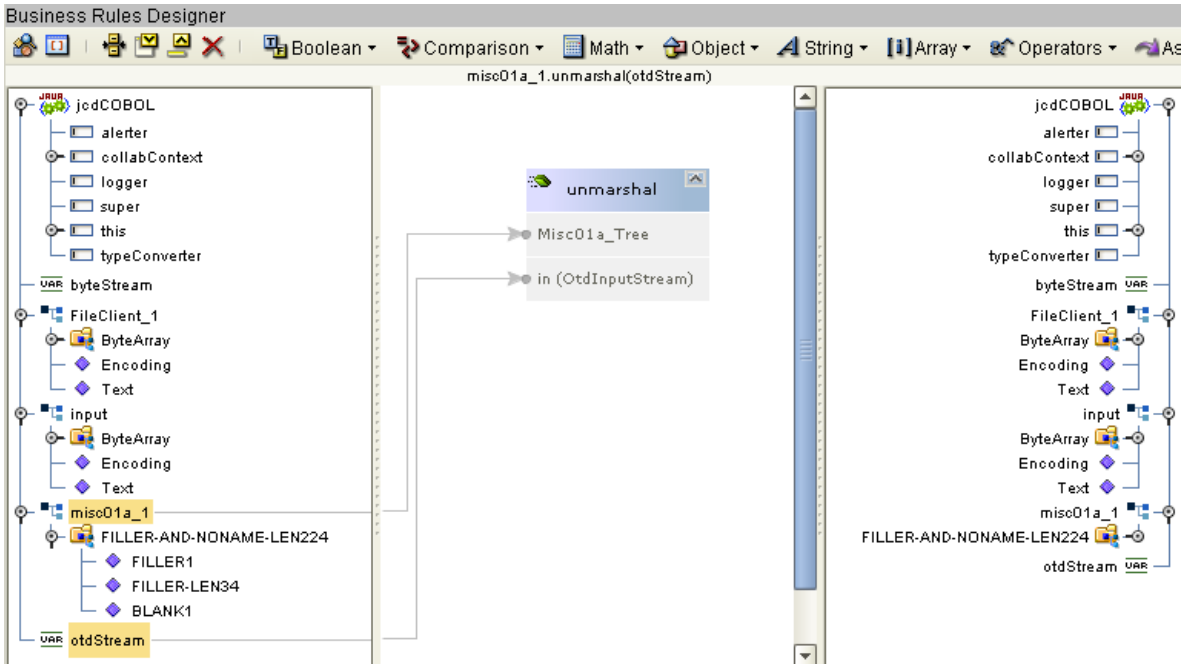
The **Copy new SimpleOtdInputStreamImpl(byte Stream) to variable otdStream** Variable is displayed in Figure 13.

Figure 13 prjCOBOL_JCD_Sample Business Rule 2



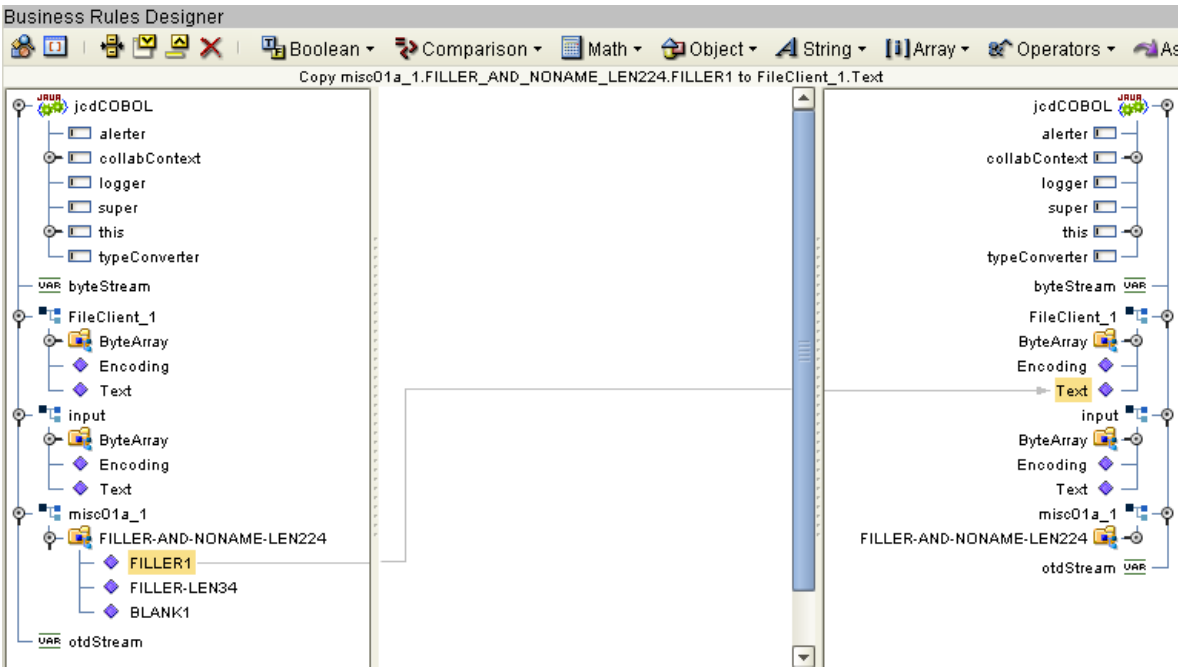
The **misc01a_1.unmarshal(otdStream)** Business Rule is displayed in Figure 14.

Figure 14 prjCOBOL_JCD_Sample Business Rule 3



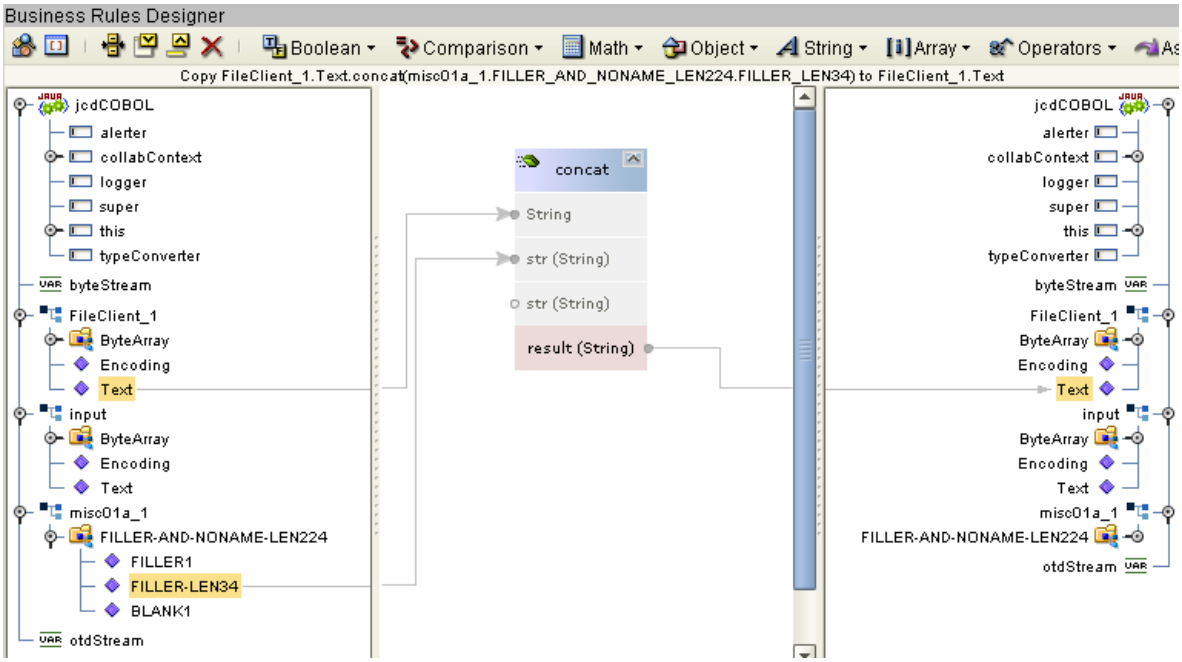
The **Copy misc01a_1.FILLER_AND_NONAME_LEN224.FILLER1 to FileClient_1.Text** Business Rule is displayed in Figure 15.

Figure 15 prjCOBOL_JCD_Sample Business Rule 4



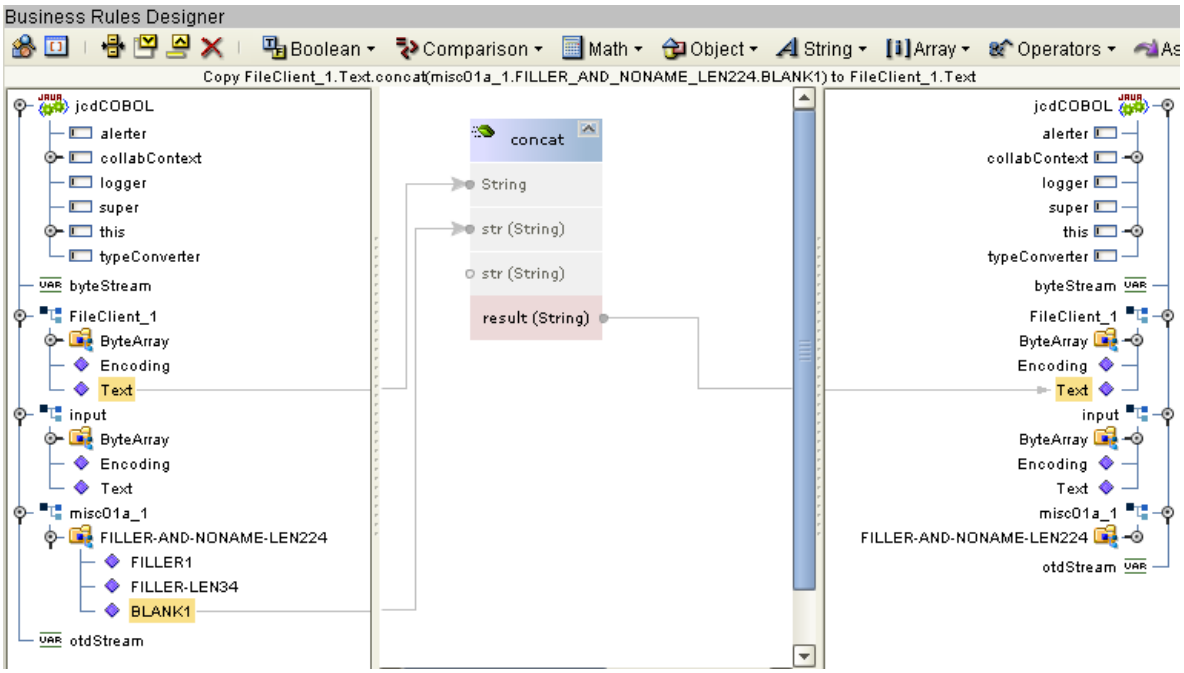
The **Copy FileClient_1.Text.concat(misc01a_1.FILLER_AND_NONAME_LEN224.FILLER_LEN34) to FileClient_1.Text** Business Rule is displayed in Figure 16.

Figure 16 prjCOBOL_JCD_Sample Business Rule 5



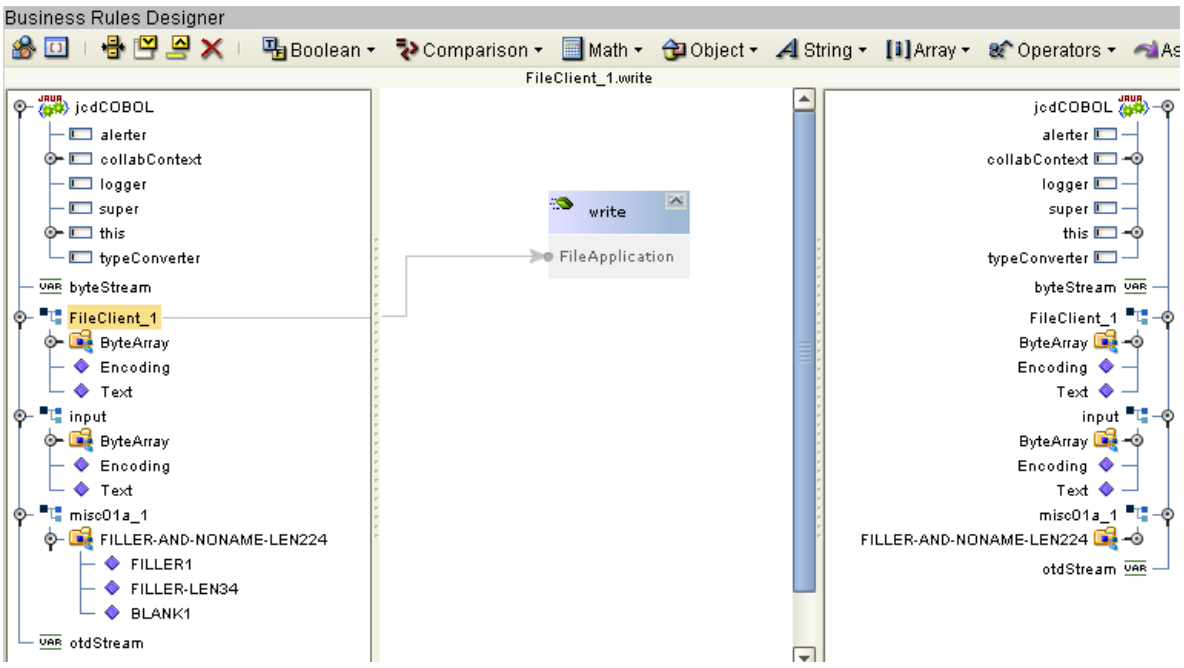
The **Copy FileClient_1.Text.concat(misc01a_1.FILLER_AND_NONAME_LEN224.BLANK1) to FileClient_1.Text** Business Rule is displayed in Figure 17.

Figure 17 prjCOBOL_JCD_Sample Business Rule 6



The **FileClient_1.write** Business Rule is displayed in Figure 18.

Figure 18 prjCOBOL_JCD_Sample Business Rule 7



Sample code from the prjCOBOL_JCD_Sample Collaboration includes the following:

```
package prjCOBOL_JCD_Sample;

public class jcdCOBOL
{
```

```
    public com.stc.codegen.logger.Logger logger;
    public com.stc.codegen.alerter.Alerter alerter;
    public com.stc.codegen.util.CollaborationContext collabContext;
    public com.stc.codegen.util.TypeConverter typeConverter;

    public void receive(
com.stc.connector.appconn.file.FileTextMessage input,
com.stc.connector.appconn.file.FileApplication FileClient_1,
Misc01a.Misc01a_Tree misc01a_1 )
        throws Throwable
    {
        // @map:java.io.ByteArrayInputStream byteStream = new
java.io.ByteArrayInputStream( input.getByteArray() );
        java.io.ByteArrayInputStream byteStream = new
java.io.ByteArrayInputStream( input.getByteArray() );
        // @map:com.stc.otd.runtime.OtdInputStream otdStream = new
com.stc.otd.runtime.provider.SimpleOtdInputStreamImpl( byteStream );
        com.stc.otd.runtime.OtdInputStream otdStream = new
com.stc.otd.runtime.provider.SimpleOtdInputStreamImpl( byteStream );
        // misc01a_1.unmarshal(otdStream)
        misc01a_1.unmarshal( otdStream );
        // Copy FILLER1 to Text
        FileClient_1.setText(
misc01a_1.getFILLER_AND_NONAME_LEN224().getFILLER1() );
        // Copy Text.concat(FILLER_LEN34) to Text
        FileClient_1.setText( FileClient_1.getText().concat(
misc01a_1.getFILLER_AND_NONAME_LEN224().getFILLER_LEN34() ) );
        // Copy Text.concat(BLANK1) to Text
        FileClient_1.setText( FileClient_1.getText().concat(
misc01a_1.getFILLER_AND_NONAME_LEN224().getBLANK1() ) );
        // FileClient_1.write
        FileClient_1.write();
    }

}
```

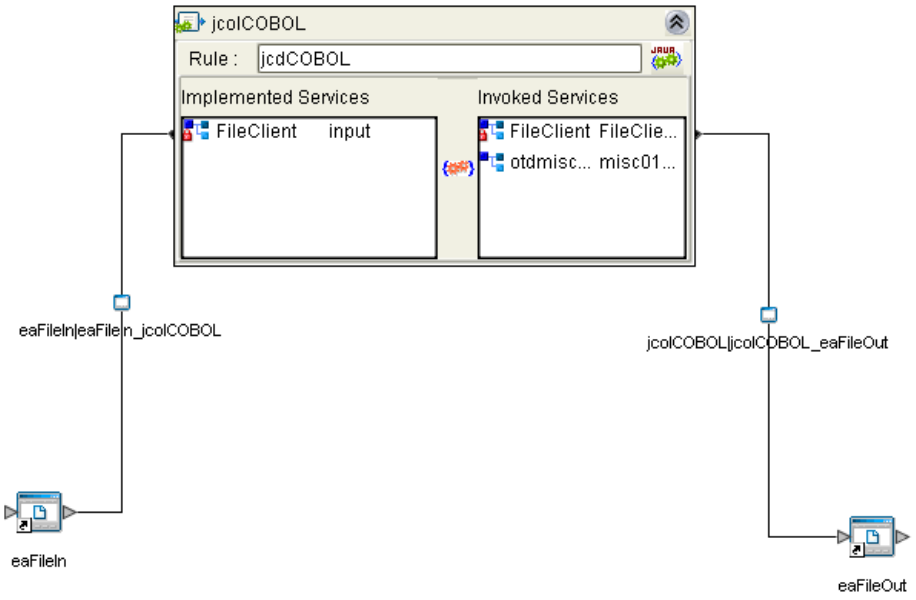
4.4.6 Binding the eWay Components

The final step in creating a Connectivity Map is binding the eWay components together.

Steps required to bind eWay components together:

1. Double-click the Connectivity Map in the Project Explorer tree. The **cmCOBOL_JCD** Connectivity Map appears in the Enterprise Designers canvas.
2. Drag and drop the **jcdCOBOL** Collaboration from the Project Explorer to the **cmCOBOL_JCD** Service. The Service icon "gears" change from red to green.
3. Double-click the **jcdCOBOL** Service. The **jcdCOBOL** Binding dialog box appears.
4. Map the input **FileClient** (under Implemented Services) to the **FileClientIN** (File) External Application. To do this, click on **FileSender** in the **jcdCOBOL** Binding dialog box, and drag the cursor to the **FileClientIN** External Application in the Connectivity Map. A link is now visible between **FileClientIN** and **eaFileIn**.
5. From the **jcdCOBOL** Binding dialog box, map **otdmis01a** (under Invoked Services) to the **eaFileOut** External Application.
6. From the **jcdCOBOL** Binding dialog box, map **FileClient_1** to the **FileClientOUT** External Application, as seen in Figure 19.

Figure 19 Connectivity Map - Associating (Binding) the Project's Components



7 Minimize the **jcdCOBOL** Binding dialog box by clicking the chevrons in the upper-right corner.

8 Save your current changes to the Repository.

4.4.7 Creating an Environment

Environments include the external systems, Logical Hosts, Integration Servers and message servers used by a Project and contain the configuration information for these components. Environments are created using the Enterprise Designer's Environment Editor.

Steps required to create an Environment:

1 From the Enterprise Designer's Enterprise Explorer, click the **Environment Explorer** tab.

2 Right-click the Repository and select **New Environment**. A new Environment is added to the Environment Explorer tree.

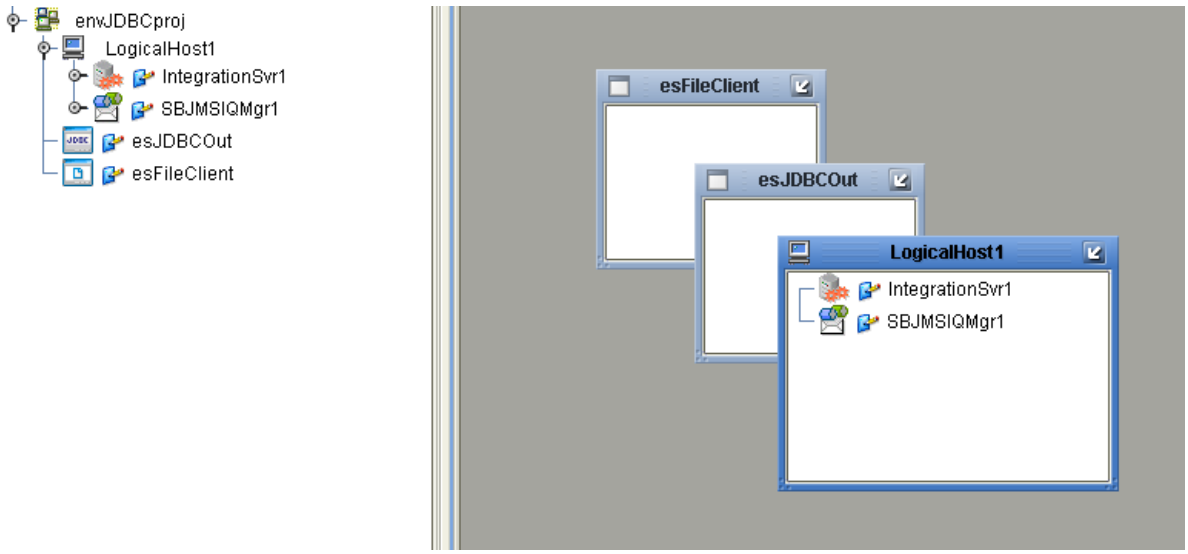
3 Rename the new Environment to **envCOBOLProj**.

4 Right-click **envCOBOLProj** and select **New > File External System**. Name the External System **esFileClient**. Click **OK**. **esFileClient** is added to the Environment Editor.

5 Right-click **envCOBOLProj** and select **New > Logical Host**. The **LogicalHost1** box is added to the Environment and **LogicalHost1** is added to the Environment Editor tree.

6 Right-click **LogicalHost1** and select **New > Sun SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under **LogicalHost1**. See Figure 20.

Figure 20 Environment Editor - envCOBOLProj



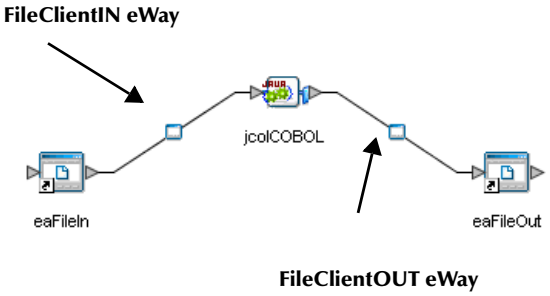
7 Save your current changes to the Repository.

4.4.8 Configuring the eWays

eWays facilitate communication and movement of data between the external applications and the eGate system. Each Connectivity Map in the **prjCOBOL_JCD_Sample** sample Project uses two eWays that are represented as nodes between the External Applications and the Business Process. See Figure 21.

You must configure eWay properties in both the Connectivity Map and the Environment Explorer.

Figure 21 eWays in the cmDelete Connectivity Map



Configuring the eWay Properties

Steps required to configure the eWay properties:

1. Double-click the **FileClientIN** eWay on the Connectivity Map and modify the properties for your system. Click **OK** to close the Properties Editor.
2. Double-click the **FileClientOUT** eWay on the Connectivity Map and modify the properties for your system. Click **OK** to close the Properties Editor.

Configuring the Environment Explorer Properties

Steps required to configure the Environment Explorer properties:

1. From the **Environment Explorer** tree, right-click the File External System (**esFileClient** in this sample), and select **Properties**. The Properties Editor opens to the File eWay Environment configuration.
2. Modify the File eWay Environment configuration properties for your system, as seen in Table 20, and click **OK**.

Table 20 File eWay Environment Properties

Section	Property Name	Required Values
Configuration > Inbound File eWay > Parameter Settings	Directory	Enter the directory that contains the input file (trigger file included in the sample Project). In this sample Project, the input file is **inputCOBOLJCD.txt.~in**.
Configuration > Outbound File eWay > Parameter Settings	Directory	Enter the directory where output files are written. In this sample Project, the output file is **COBOL_JCDoutput1.dat**.

Configuring the Integration Server

You must set your SeeBeyond Integration Server Password property before deploying your Project.

1. From the Environment Explorer, right-click **IntegrationSvr1** under your **Logical Host**, and select **Properties** from the shortcut menu. The Integration Server Properties Editor appears.
2. Click the **Password** property field under **Sun SeeBeyond Integration Server Configuration**. An ellipsis appears in the property field.
3. Click the ellipsis. The **Password Settings** dialog box appears.
4. Enter **STC** as the **Specific Value** and as the **Confirm Password**, and click **OK**.
5. Click **OK** to accept the new property and close the Properties Editor.

For more information on deploying a Project see the *Sun SeeBeyond Java™ Composite Application Platform Suite Deployment Guide*.

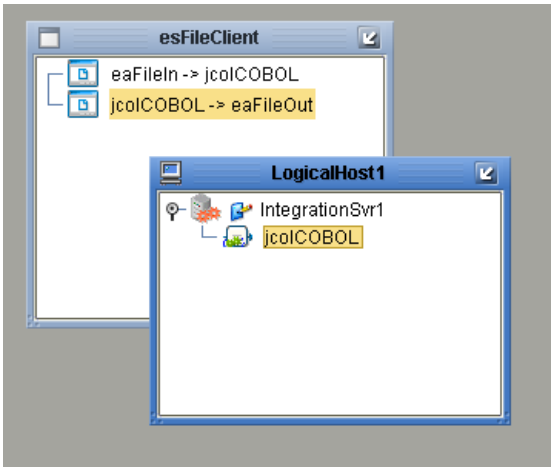
4.4.9 Creating the Deployment Profile

A Deployment Profile is used to assign services and message destinations to the Integration Server and message server. Deployment profiles are created using the Deployment Editor.

Steps required to create the Deployment Profile:

1. From the Enterprise Explorer's Project Explorer, right-click the **prjCOBOL_JCD_Sample** Project and select **New** > **Deployment Profile**.
2. Enter a name for the Deployment Profile (for this sample **dpCOBOL_JCD**). Select **envCOBOLProj** as the Environment and click **OK**.
3. From the Deployment Editor toolbar, click the **Automap** icon. The Project's components are automatically mapped to their system windows, as seen in Figure 22.

Figure 22 Deployment Profile



4.4.10 Creating and Starting the Domain

To build and deploy your Project, you must first create a domain. A domain is an instance of a Logical Host. After the domain is created, the Project is built and then deployed.

Note: *You are only required to create a domain once when you install the Sun Java Composite Application Platform Suite.*

Steps required to create and start the domain:

1. Navigate to your **<JavaCAPS51>\logicalhost** directory (where <JavaCAPS51> is the location of your Sun Java Composite Application Platform Suite installation).
2. Double-click the **domainmgr.bat** file. The **Domain Manager** appears.
3. If you have already created a domain, select your domain in the Domain Manager and click the **Start an Existing Domain** button. Once your domain is started, a green check mark indicates that the domain is running.

4 If there are no existing domains, a dialog box indicates that you can create a domain now. Click **Yes**. The **Create Domain** dialog box appears.

5 Make any necessary changes to the **Create Domain** dialog box and click **Create**. The new domain is added to the Domain Manager. Select the domain and click the **Start an Existing Domain** button. Once your domain is started, a green check mark indicates that the domain is running.

For more information about creating and managing domains see the *eGate Integrator System Administration Guide*.

4.4.11 Building and Deploying the Project

The Build process compiles and validates the Project's Java files and creates the Project EAR file.

Build the Project

1 From the Deployment Editor toolbar, click the **Build** icon.

2 If there are any validation errors, a **Validation Errors** pane will appear at the bottom of the Deployment Editor and displays information regarding the errors. Make any necessary corrections and click **Build** again.

3 After the Build has succeeded you are ready to deploy your Project.

Deploy the Project

1 From the Deployment Editor toolbar, click the **Deploy** icon. Click **Yes** when the **Deploy** prompt appears.

2 A message appears when the project is successfully deployed. You can now test your sample.

4.4.12 Running the Sample

Additional steps are required to run the deployed sample Project.

Steps required to run the sample Project:

1 Rename one of the trigger files included in the sample Project from **<filename>.in.~in** to **<filename>.in** to run the corresponding operation.

The File eWay polls the directory every five seconds for the input file name (as defined in the Inbound File eWay Properties window). The COBOL JCD then transforms the data, and the File eWay sends the output to an Output file name (as defined in the outbound File eWay Properties window).

2 Verify the output data by viewing the sample output files. See **About the COBOL Copybook Converter Sample Projects** on page 32 for more details on the types of output files used in this sample Project. The output files may change depending on the number of times you execute the sample Project and the content of the input files.

4.5 Building, Deploying, and Running the prjCOBOL_BP_Sample Project

This section provides step-by-step instructions for manually creating the prjCOBOL_BP_Sample sample Project.

Steps required to create the sample project

4.5.1 Creating a Project

The first step is to create a new Project in the Enterprise Designer.

1. Start the Enterprise Designer.
2. From the Project Explorer tree, right-click the Repository and select **New Project**. A new Project (**Project1**) appears on the Project Explorer tree.
3. Click twice on **Project1** and rename the Project (for this sample, **prjCOBOL_BP_Sample**).

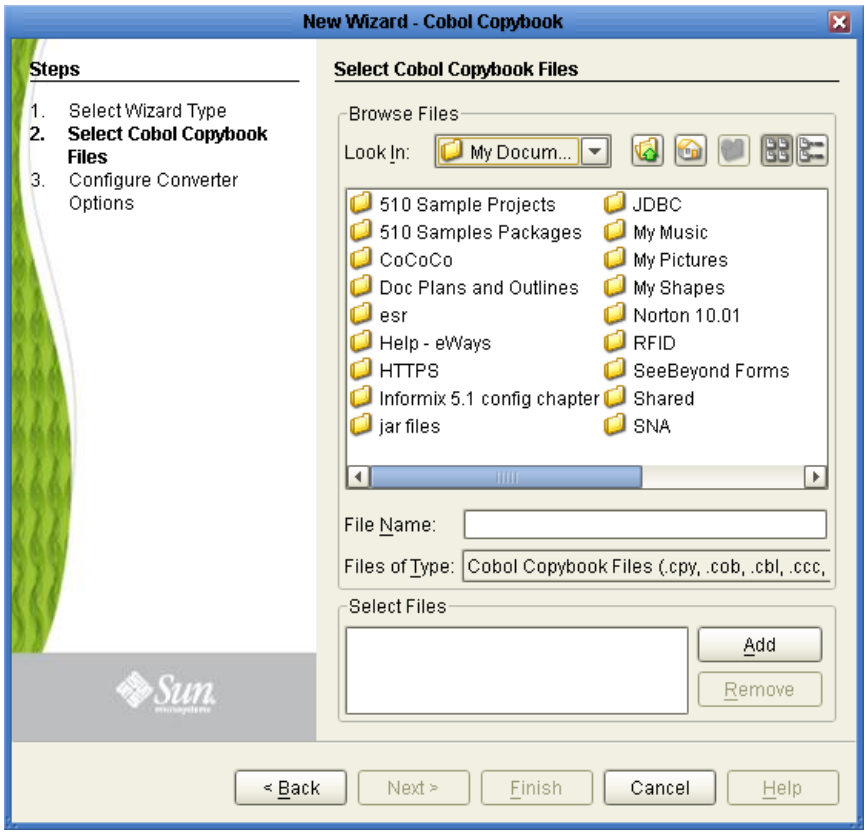
4.5.2 Creating the OTDs

The sample Project requires an OTD to interact with the COBOL Copybook Converter. The OTD for the BPEL sample Project is built on the sample COBOL file **qan3glr1.cobol**.

Steps required to create the COBOL qan3glr1 OTD:

1. Right-click your new Project in the Enterprise Designer's Project Explorer, and select **New > Object Type Definition**.
2. Select **COBOL Copybook** from the list of OTD Wizards and click **Next**. The **Select COBOL Copybook Files** page appears.

Figure 23 COBOL Copybook Wizard—COBOL Copybook Selection



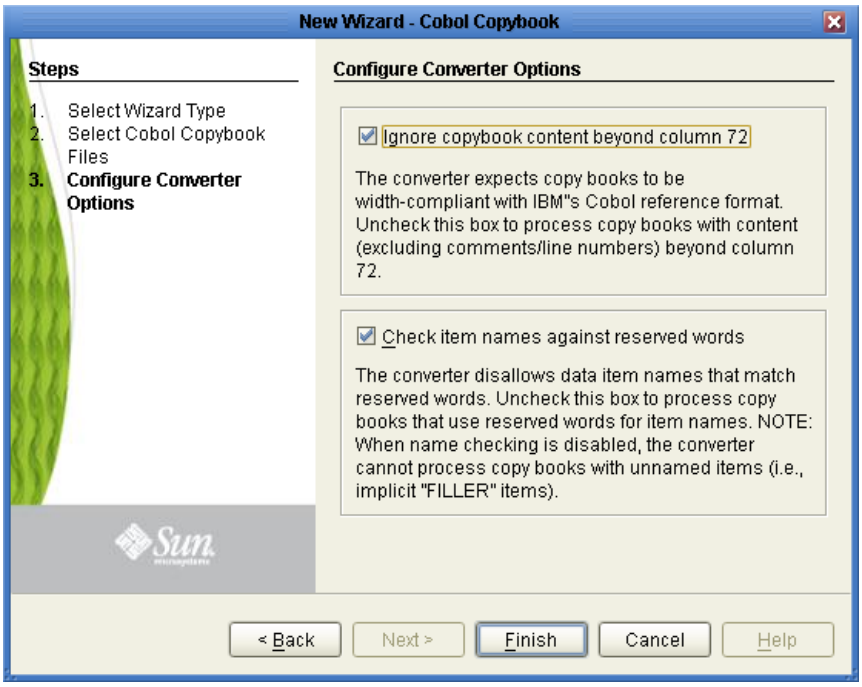
3 Browse for the desired COBOL Copybook file and highlight it.

4 Click the **Add** button to include a copybook file in a project.

5 Repeat Steps 3 and 4 for each file to include in the project. To remove a copybook file from the project, highlight the file name in the **Select Files** container and click **Remove**.

6 Click **Next**. The **Configure Converter Options** page appears.

Figure 24 COBOL Copybook Wizard—Configure Converter Options



7 Optionally, add/remove checks from boxes to enable/disable options:

- **Ignore copybook content beyond column 72** -- The Converter expects copybooks to be width-compliant with IBM's COBOL reference format. Uncheck this box to process books with content (excluding comments/line numbers) beyond column 72. Default: enabled (box is checked).
- **Check Item names against reserved words** -- The Converter disallows data item names that match reserved words. Uncheck this box to process copy books that use reserve words for item names. When name checking is disabled, the Converter cannot process copy books with unnamed items (i.e., implicit 'FILLER' items). Default: enabled (box is checked).

8 Click **Finish**. The **OTD Editor** window appears, displaying the OTD.

4.5.3 Creating the Business Process

Steps required to create the Business Process include:

- Creating the business process flow
- Configuring the modeling elements

Creating the Business Process Flow

The business process flow contains all the BPEL elements that make up a business process.

Steps to create a business process flow include:

1. Right-click your new Project in the Enterprise Designer's Project Explorer, and select **New** > **Business Process** from the shortcut menu. The eInsight Business Process Designer appears and **BusinessProcess1** is added to the Project Explorer tree. Rename **BusinessProcess1** to **BP1**.
2. Add the following activities to the Business Process Designer canvas.

Table 21 Business Process Activities

Business Process	Activity
BP1	FileClient.Receive qan3glr1.unmarshal FileClient.Write

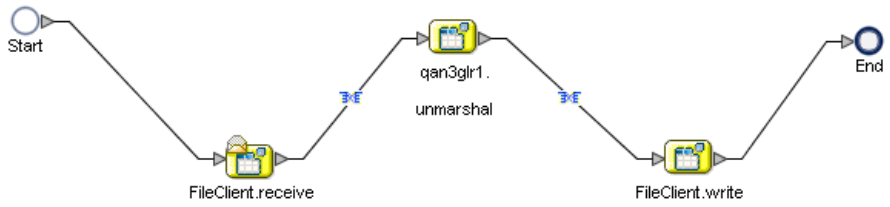
Configuring the BP1 Modeling Elements

Business Rules, created between the Business Process Activities, allow you to configure the relationships between the input and output Attributes of the Activities using the Business Process Designer's Business Rule Designer.

Once you have connected the modeling elements together, begin adding the business processes necessary to facilitate the operation. See Figure 25 for an illustration of how all the modeling elements appear when connected.

Note: *Review the eInsight Business Process Manager User's Guide for a more detailed description of the steps required to connect and add business rules to a modeling elements in a business process.*

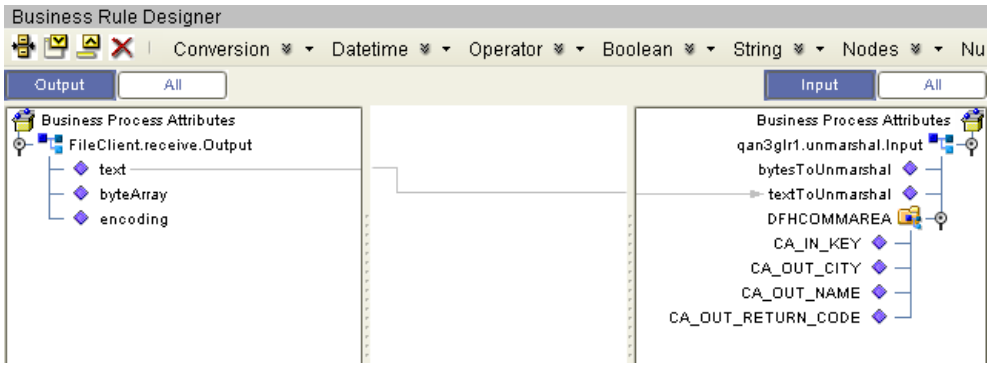
Figure 25 prjCOBOL_BP_Sample Business Process



Steps required to configure the BP1 business process:

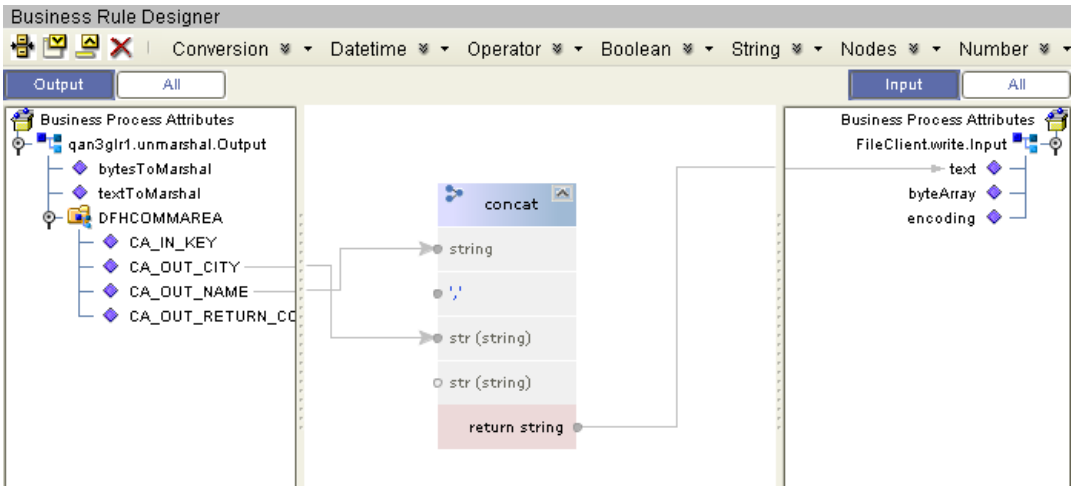
1. Configure the business rule between the **FileClient.receive** and **qan3glr1** Activities, as seen in Figure 26.

Figure 26 prjCOBOL_BP_Sample Business Rule 1



2 Configure the business rule between the **qan3glr1** and **FileClient.write** Activities, as seen in Figure 27.

Figure 27 prjCOBOL_BP_Sample Business Rule 2



4.5.4 Creating a Connectivity Map

The Connectivity Map provides a canvas for assembling and configuring a Project's components.

Steps required to create a new Connectivity Map:

1 From the Project Explorer tree, right-click the new **prjCOBOL_BP_Sample** Project and select **New > Connectivity Map** from the shortcut menu.

2 The New Connectivity Map appears and a node for the Connectivity Map is added under the Project, on the Project Explorer tree labeled **CMap1**.

The icons in the toolbar represent the available components used to populate the Connectivity Map canvas.

Populating the Connectivity Map

Add the Project components to the Connectivity Map by dragging the icons from the toolbar to the canvas.

Each Connectivity Map in the **prjCOBOL_BP_Sample** sample Project requires the following components:

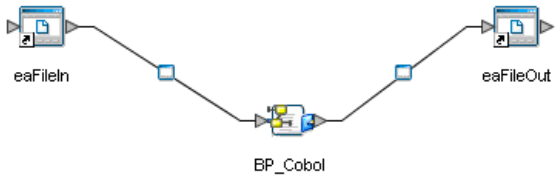
- File External Application (2)
- Service

Any eWay added to the Connectivity Map is associated with an External System.

Steps required to select a File External System:

1. Click the **External Application** icon on the Connectivity Map toolbar.
2. Select the external systems necessary to create your Project (for this sample, **File**). Icons representing the selected external system are added to the Connectivity Map toolbar.
3. Rename the following components and then save changes to the Repository:
   - File1 to eaFileIN
   - File2 to eaFileOUT

Figure 28 COBOL Copybook BPEL Sample Connectivity Map



To Select a COBOL Business Process

1. Drag a business process from the Enterprise Explorer Project Explorer onto the corresponding Connectivity Map. For example, drag the **BP1** business process onto the **cmCOBOL_BP** Connectivity Map.
2. Save your changes to the Repository

Binding the eWay Components

The final step in creating a Connectivity Map is binding the eWay components together.

Steps required to bind eWay components together:

1. Double-click the Connectivity Map in the Project Explorer tree. The **cmCOBOL_BP** Connectivity Map appears in the Enterprise Designers canvas.
2. Drag and drop the **BP1** Business Process from the Project Explorer to the **BP_COBOL** Service. The Service icon "gears" change from red to green.

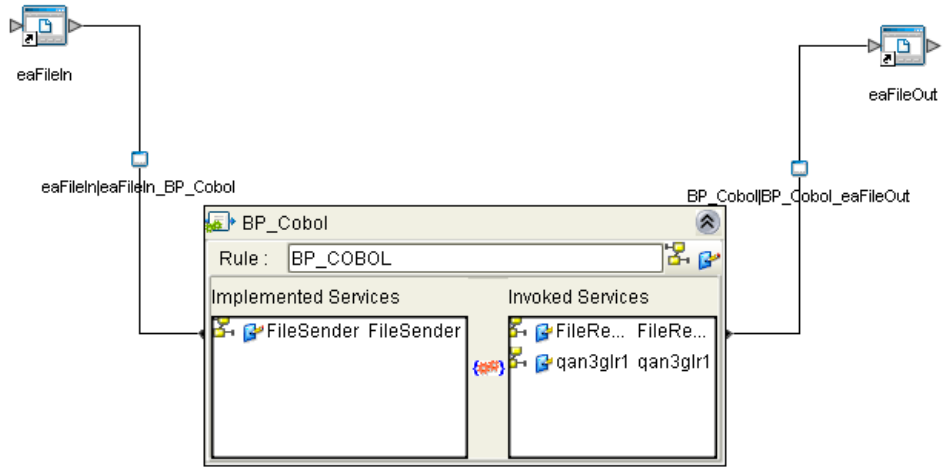
3 Double-click the **BP_COBOL** Service. The **BP_COBOL** Binding dialog box appears.

4 Map the input **FileClient** (under Implemented Services) to the **FileClientIN** (File) External Application. To do this, click on **FileSender** in the **BP_COBOL** Binding dialog box, and drag the cursor to the **FileClientIN** External Application in the Connectivity Map. A link is now visible between **FileClientIN** and **eaFileIn**.

5 From the **BP_COBOL** Binding dialog box, map **qan3glr1** (under Invoked Services) to the **eaFileOut** External Application.

6 From the **BP_COBOL** Binding dialog box, map **FileClient_1** to the **FileClientOUT** External Application, as seen in Figure 29.

Figure 29 Connectivity Map - Associating (Binding) the Project's Components



7 Minimize the **BP_COBOL** Binding dialog box by clicking the chevrons in the upper-right corner.

8 Save your current changes to the Repository.

4.5.5 Creating an Environment

Environments include the external systems, Logical Hosts, Integration Servers and message servers used by a Project and contain the configuration information for these components. Environments are created using the Enterprise Designer's Environment Editor.

Steps required to create an Environment:

1 From the Enterprise Designer's Enterprise Explorer, click the **Environment Explorer** tab.

2 Right-click the Repository and select **New Environment**. A new Environment is added to the Environment Explorer tree.

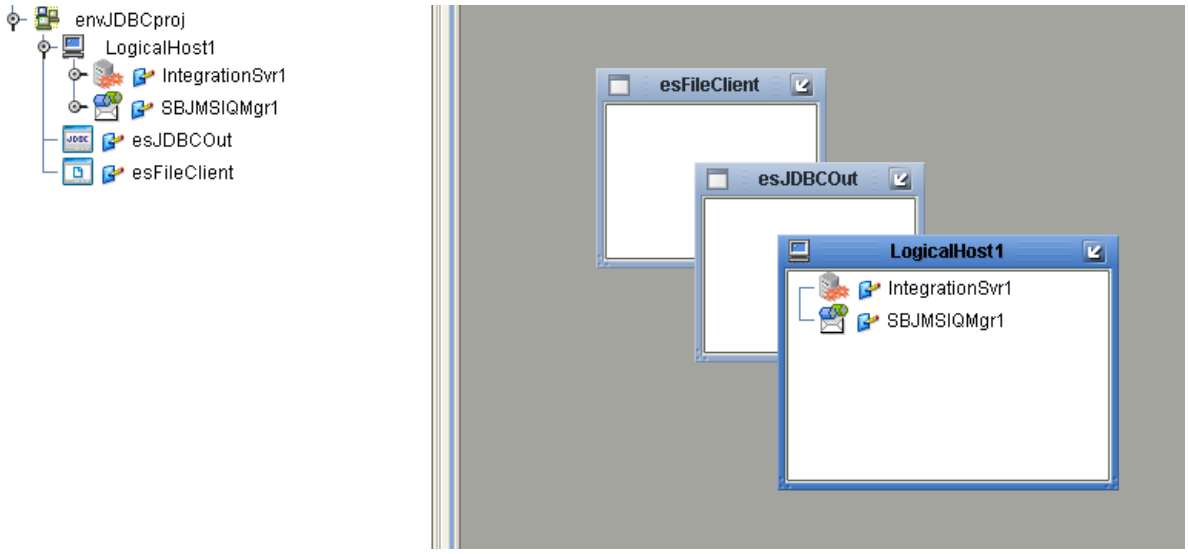
3 Rename the new Environment to **envCOBOLProj**.

4 Right-click **envCOBOLProj** and select **New > File External System**. Name the External System **esFileClient**. Click **OK**. **esFileClient** is added to the Environment Editor.

5 Right-click **envCOBOLProj** and select **New > Logical Host**. The **LogicalHost1** box is added to the Environment and **LogicalHost1** is added to the Environment Editor tree.

6 Right-click **LogicalHost1** and select **New > Sun SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under **LogicalHost1**. See Figure 30.

Figure 30 Environment Editor - envCOBOLProj



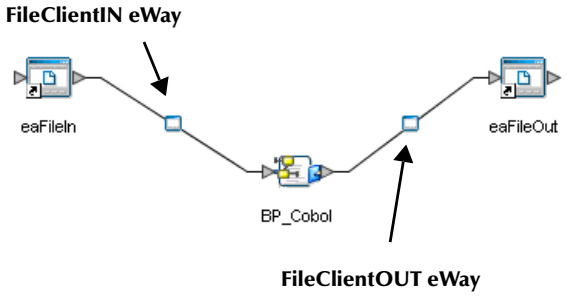
7 Save your current changes to the Repository.

4.5.6 Configuring the eWays

eWays facilitate communication and movement of data between the external applications and the eGate system. Each Connectivity Map in the **prjCOBOL_BP_Sample** sample Project uses two eWays that are represented as nodes between the External Applications and the Business Process. See Figure 31.

You must configure eWay properties in both the Connectivity Map and the Environment Explorer.

Figure 31 eWays in the cmDelete Connectivity Map



Configuring the eWay Properties

Steps required to configure the eWay properties:

1. Double-click the **FileClientIN** eWay on the Connectivity Map and modify the properties for your system. Click **OK** to close the Properties Editor.
2. Double-click the **FileClientOUT** eWay on the Connectivity Map and modify the properties for your system. Click **OK** to close the Properties Editor.

Configuring the Environment Explorer Properties

Steps required to configure the Environment Explorer properties:

1. From the **Environment Explorer** tree, right-click the File External System (**esFileClient** in this sample), and select **Properties**. The Properties Editor opens to the File eWay Environment configuration.
2. Modify the File eWay Environment configuration properties for your system, as seen in Table 22, and click **OK**.

Table 22 File eWay Environment Properties

Section	Property Name	Required Values
Configuration > Inbound File eWay > Parameter Settings	Directory	Enter the directory that contains the input file (trigger file included in the sample Project). In this sample Project, the input file is **inputCOBOLBP.txt.~in**.
Configuration > Outbound File eWay > Parameter Settings	Directory	Enter the directory where output files are written. In this sample Project, the output file is **COBOLBPoutput0.dat**.

Configuring the Integration Server

You must set your SeeBeyond Integration Server Password property before deploying your Project.

1. From the Environment Explorer, right-click **IntegrationSvr1** under your **Logical Host**, and select **Properties** from the shortcut menu. The Integration Server Properties Editor appears.
2. Click the **Password** property field under **Sun SeeBeyond Integration Server Configuration**. An ellipsis appears in the property field.
3. Click the ellipsis. The **Password Settings** dialog box appears.
4. Enter **STC** as the **Specific Value** and as the **Confirm Password**, and click **OK**.
5. Click **OK** to accept the new property and close the Properties Editor.

For more information on deploying a Project see the *Sun SeeBeyond Java™ Composite Application Platform Suite Deployment Guide*.

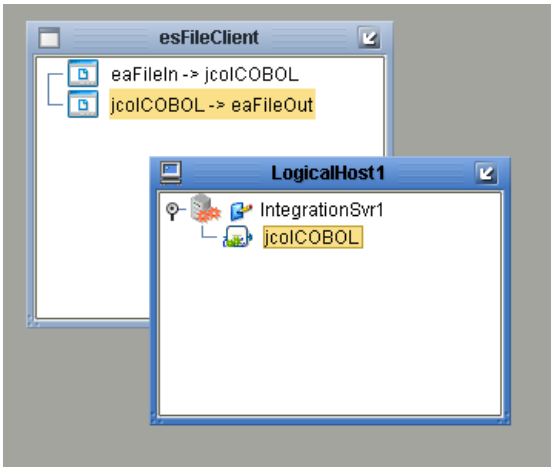
4.5.7 Creating the Deployment Profile

A Deployment Profile is used to assign services and message destinations to the Integration Server and message server. Deployment profiles are created using the Deployment Editor.

Steps required to create the Deployment Profile:

1. From the Enterprise Explorer's Project Explorer, right-click the **prjCOBOL_BP_Sample** Project and select **New** > **Deployment Profile**.
2. Enter a name for the Deployment Profile (for this sample **dpCOBOL_BP**). Select **envCOBOLProj** as the Environment and click **OK**.
3. From the Deployment Editor toolbar, click the **Automap** icon. The Project's components are automatically mapped to their system windows, as seen in Figure 32.

Figure 32 Deployment Profile



4.5.8 Creating and Starting the Domain

To build and deploy your Project, you must first create a domain. A domain is an instance of a Logical Host. After the domain is created, the Project is built and then deployed.

Note: *You are only required to create a domain once when you install the Sun Java Composite Application Platform Suite.*

Steps required to create and start the domain:

1. Navigate to your **<JavaCAPS51>\logicalhost** directory (where <JavaCAPS51> is the location of your Sun Java Composite Application Platform Suite installation).
2. Double-click the **domainmgr.bat** file. The **Domain Manager** appears.
3. If you have already created a domain, select your domain in the Domain Manager and click the **Start an Existing Domain** button. Once your domain is started, a green check mark indicates that the domain is running.
4. If there are no existing domains, a dialog box indicates that you can create a domain now. Click **Yes**. The **Create Domain** dialog box appears.
5. Make any necessary changes to the **Create Domain** dialog box and click **Create**. The new domain is added to the Domain Manager. Select the domain and click the **Start an Existing Domain** button. Once your domain is started, a green check mark indicates that the domain is running.

For more information about creating and managing domains see the *eGate Integrator System Administration Guide*.

4.5.9 Building and Deploying the Project

The Build process compiles and validates the Project's Java files and creates the Project EAR file.

Build the Project

1. From the Deployment Editor toolbar, click the **Build** icon.
2. If there are any validation errors, a **Validation Errors** pane will appear at the bottom of the Deployment Editor and displays information regarding the errors. Make any necessary corrections and click **Build** again.
3. After the Build has succeeded you are ready to deploy your Project.

Deploy the Project

1. From the Deployment Editor toolbar, click the **Deploy** icon. Click **Yes** when the **Deploy** prompt appears.
2. A message appears when the project is successfully deployed. You can now test your sample.

4.5.10 Running the Sample

Additional steps are required to run the deployed sample Project.

Steps required to run the sample Project:

1. Rename one of the trigger files included in the sample Project from **<filename>.in.~in** to **<filename>.in** to run the corresponding operation.

   The File eWay polls the directory every five seconds for the input file name (as defined in the Inbound File eWay Properties window). The COBOL JCD then transforms the data, and the File eWay sends the output to an Output file name (as defined in the outbound File eWay Properties window).

2. Verify the output data by viewing the sample output files. See **About the COBOL Copybook Converter Sample Projects** on page 32 for more details on the types of output files used in this sample Project. The output files may change depending on the number of times you execute the sample Project and the content of the input files.

Appendix A

Frequently Asked Questions

This appendix contains Frequently Asked Questions (FAQs) about the COBOL Copybook Converter. They are intended to be a quick reference for issues not discussed elsewhere in this *User's Guide*. The FAQs are sorted by bolded **keywords** in the FAQs column in Table 23.

Table 23 Frequently Asked Questions

FAQ	Answer
Can I use **ASCII** data to process Cobol Copybook?	Yes, if the data is only in **PIC'X'** or **PIC'9'** format. If the data is zoned decimal like **COMP-3**, the Converter will fail since there is no translation from EBCDIC to ASCII for this kind of data.
How does the Converter support **Binary** data?	The Converter supports numeric binary data according to its USAGE specification. The Converter does not apply encoding to binary data during marshal, nor does it apply decoding to binary data during unmarshal.
Does the Converter support **COMP** fields?	Yes, it does.
Does the Converter support **copy statements** within the copybook?	No.
What is the limitation with **edited** field?	Edited fields accept String data of a length equal to the storage width defined by the picture, with alphanumeric truncation and padding rules applied as necessary. The data itself is not validated against the edited item's picture.
Does the Converter support **Encoding**?	Yes, it supports Encoding via standard Java Standard API Encoding support. Java itself may not support all encoding, but it supports the majority of those commonly used.
Can the Converter **initialize** data in the copybook?	Numeric fields will be initialized to zero. Character fields will be initialized to blanks. This is the default behavior. There is an option to initialize data to **HIGH** or **LOW**. **resetHIGH** will always initialize to all "1"s. **resetLOW** will always initialize to all "0"s. This is regardless of numeric or character field.

Table 23 Frequently Asked Questions (Continued)

FAQ	Answer
Does the Converter support **marshaling** and **unmarshaling** in BPEL?	Yes, it supports both operations.
Does the Converter support **OCCURS**?	Yes, it does. However, if the number of occurrences is zero, the field will not show up in the OTD Tester. While you may interpret this absence as an error, it is not.
Would the **OTD Tester** work with ASCII data?	No. The default for the Converter is EBCDIC.
Is there any limitation with the **OTD Tester**?	Binary data will not be displayed properly.
How does **padding** work?	If a given Cobol OTD field is specified to hold character data, and the data moved into it is less than its maximum size, the data is space-padded, left-justified. If the data is binary (numeric), the data is zero-padded.
Can the Converter **parse** a COBOL program and just pick up the copybook from there?	No, the Converter is not a COBOL parser. It is only designed to process Copybook Data Division data description entries.
How does the Converter handle poor **performance** if the copybook is too big?	The Converter has a parameter which allows you to set validation to **ON** or **OFF**. When it is set to **OFF**, validation will not be performed. The default is **OFF**.
Does the COBOL Copybook Converter support **Redefine**?	Yes, but the Converter is set to use the root level format to marshal and unmarshal data. The Converter has no way of knowing which **Redefine** setting you are using.
Does the COBOL Copybook Converter support leading and trailing **SIGN** clauses?	Yes, the Converter supports the SIGN clause. The support forms are the following: ▪ SIGN IS LEADING ▪ SIGN IS LEADING SEPARATE ▪ SIGN IS TRAILING ▪ SIGN IS TRAILING SEPARATE When the SIGN IS LEADING or SIGN IS TRAILING form is attached to a Copybook entry, it indicates that the corresponding OTD field interprets the value it contains as having SIGN information stored as part of the first *digit*. When the SEPARATE phrase is specified, it indicates the sign information is stored in the first *character*.
Is there any limit to the **size** of the data element?	Yes, the Converter does not support fields greater than 96K in size.
Does the Converter support **truncation** if the field is longer than the one in the copybook?	It will truncate character strings as well as numeric fields.

Index