SeeBeyond ICAN Suite

Cobol Copybook Converter User's Guide

Release 5.0.4

SEEBEYOND®

Version 20040806173101.

Contents

Chapter 1

Introduction

This user's guide describes how to use the Cobol Copybook Converter to convert input data to COBOL copybook specifications.

In This Chapter

- **Contents of This Guide** on page 5
- **Writing Conventions** on page 5
- **Supporting Documents** on page 7
- **SeeBeyond Web Site** on page 7

1.1 Contents of This Guide

This guide contains the following information:

- **Chapter 2, "Introducing the Cobol Copybook Converter" on page 8** provides an overview of the Cobol Copybook Converter.
- **Chapter 3, "Installing the Cobol Copybook Converter" on page 10** describes how to install the Cobol Copybook Converter and its sample Project.
- **Chapter 5, "Locating, Importing, and Using the Sample Projects" on page 21** describes how to use the Cobol Copybook Converter. The chapter also includes procedures for importing and using the Cobol Copybook sample Project.

1.2 Writing Conventions

The following writing conventions are observed throughout this document.

Table 1 Writing Conventions

Text	Convention	Example
Button, file, icon, parameter, variable, method, menu, and object names.	**Bold** text	▪ Click **OK** to save and close. ▪ From the **File** menu, select **Exit**. ▪ Select the **logicalhost.exe** file. ▪ Enter the **timeout** value. ▪ Use the **getClassName()** method. ▪ Configure the **Inbound** File eWay.
Command line arguments and code samples	`Fixed` font. Variables are shown in ***bold italic***.	`bootstrap -p` ***`password`***
Hypertext links	**Blue** text	**http://www.seebeyond.com**

Additional Conventions

Windows Systems

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

Path Name Separator

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

1.3 Supporting Documents

The following SeeBeyond documents provide additional information about the ICAN Suite:

- *SeeBeyond Integrated Composite Application Network Suite Primer*
- *SeeBeyond ICAN Suite Installation Guide*
- *eGate Integrator User's Guide*
- *eGate Integrator Tutorial*
- *SeeBeyond ICAN Suite Deployment Guide*

1.4 SeeBeyond Web Site

The SeeBeyond Web site is a useful source for product news and technical support information at **www.seebeyond.com**.

Chapter 2

Introducing the Cobol Copybook Converter

This chapter provides an overview of the Cobol Copybook Converter and includes the following sections:

In This Chapter

2.1 About the Cobol Copybook Converter

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 will neither be able to communicate with each other nor be capable of transferring data between applications and platforms.

The Cobol Copybook Converter converts copybook descriptions, and creates OTDs designed to encapsulate data conforming to the description. The generated OTD is a model; a user-friendly abstraction of the data it contains. 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.

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.*

Unsupported Features

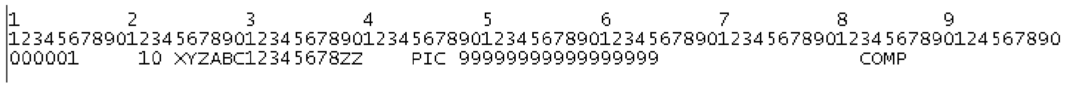
The following Cobol Copybook features are not supported by the Cobol Copybook Converter:

- **Cobol Copy Statements** — Cobol copy statements that are imbedded 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 will interpret this and create 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.

Copybooks with content beyond column 72

Copybook content 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.



```
1         2         3         4         5         6         7         8         9
12345678901234567890123456789012345678901234567890123456789012345678901234567890124567890
000001     10 XYZABC12345678ZZ    PIC 999999999999999999                COMP
```

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).

Chapter 3

Installing the Cobol Copybook Converter

This chapter describes how to install the Cobol Copybook Converter.

In This Chapter

3.1 Installation Requirements

3.1.1 Supported Operating Systems

The Cobol Copybook Converter is available for the following operating systems:

- Windows 2000

3.1.2 System Requirements

The system requirements for the Cobol Copybook Converter are the same as for eGate Integrator. For information, refer to the *eGate Integrator Installation Guide*. Additional system requirements include:

- The system where the Cobol Cobybook Converter is installed needs approximately 20 MB of free disk space for the application and its configuration, library, and script files.
- Cobol Copybook Converter 5.0.4 requires a 5.0.4 or higher version of the logical host.

3.2 Installing the Cobol Copybook Converter

During the eGate Integrator installation process, the Enterprise Manager, a web-based application, is used to select and upload products as .sar files from the eGate installation CD-ROM to the Repository.

The installation process includes installing the following components:

- Installing the Repository
- Uploading products to the Repository
- Downloading components (such as Enterprise Designer and Logical Host)
- Viewing product information home pages

Follow the instructions for installing the eGate Integrator in the *SeeBeyond ICAN Suite Installation Guide*, and include the following steps:

1. During the procedures for uploading files to the eGate Repository using the Enterprise Manager, after uploading the **eGate.sar** file, select and upload the following below as described in the *SeeBeyond ICAN Suite Installation Guide*:
    - **CobolCopyBook.sar** (to install the Cobol Copybook Converter)
    - **FileeWay.sar** (to install the File eWay, used in the sample Projects)
    - **CobolCopyBookDocs.sar** (to install the user guide and the sample Projects)
2. In the Enterprise Manager, click the **DOCUMENTATION** tab.
3. Click **Cobol Copybook Converter**.
4. In the right-hand pane, click **Download Sample**, and select a location for the .zip file to be saved.

    For information about importing and using the sample, refer to **"Locating, Importing, and Using the Sample Projects" on page 21**.

3.3 Post Installation Tasks

Upon successful completion of installing the required eWays and eWay documentation, you should open the **readme.txt** file included with the Cobol Copybook Converter downloaded sample file. This file contains information about patches or ESRs that may be required to run the Cobol Copybook Converter or to run project sample files that use the Cobol Copybook Converter.

After ensuring you have all the required Cobol Copybook converter patches or ESRs, you must then incorporate the eWay into an eGate Project and Environment in Enterprise Designer. The next chapters describe how you add the eWay to an eGate Project and an eGate Environment, how you configure the eWay and how to build the necessary OTDs.

Chapter 4

Using the Cobol Copybook Converter OTD Wizard

This chapter describes how to build the business logic for Cobol Copybook Converter Projects. Project business logic is contained in Business Processes for eInsight, and in Collaborations for eGate Integrator used without eInsight.

To build Cobol Copybook Project business logic, you use the Cobol Copybook wizard to create the Cobol Copybook Converter OTD. You then create the Business Processes or Collaborations, and the Connectivity Maps.

In This Chapter

4.1 About the Cobol Copybook Wizard

The Cobol Copybook wizard is used to create copybook converter OTDs. These OTDs can then later be used in Collaboration Definitions to create the business logic behind the Collaborations.

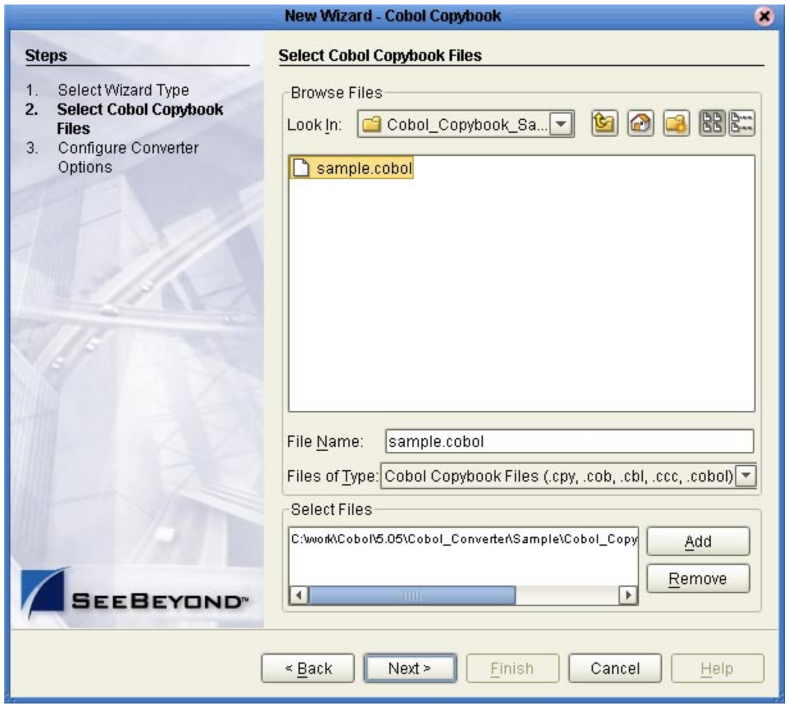
4.2 Creating Cobol Copybook OTDs

You create Cobol Copybook Converter OTDs with the Cobol Copybook wizard in the Enterprise Designer.

To create Cobol Copybook OTDs

1. In the Explorer tab of the Enterprise Designer, right-click **%Project Name% > New > Object Type Definition**. The **New Object Type Definition Wizard** dialog box appears.
2. Click **Cobol Copybook** and click **Next**. The **Select Cobol Copybook Files** page appears.

Figure 2 Cobol Copybook Wizard—Cobol Copybook Selection


New Wizard - Cobol Copybook
Steps
1. Select Wizard Type
2. Select Cobol Copybook Files
3. Configure Converter Options
Select Cobol Copybook Files
Browse Files
Look In: Cobol_Copybook_Sa...
sample.cobol
File Name: sample.cobol
Files of Type: Cobol Copybook Files (.cpy, .cob, .cbl, .ccc, .cobol)
Select Files
C:\work\Cobol\5.05\Cobol_Converter\Sample\Cobol_Copy
Add
Remove
SEEBEYOND
< Back
Next >
Finish
Cancel
Help

3 Browse for and highlight the desired Cobol Copybook file.

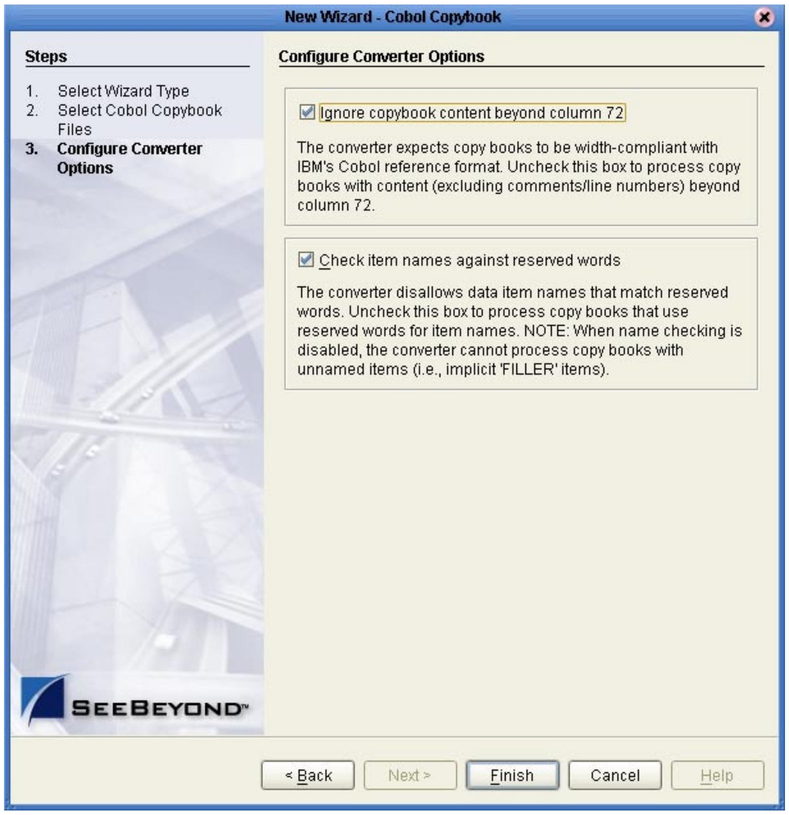
4 Click the **Add** button to include a copybook file in a project.

5 Repeat Steps 3 and 4 for each file to include in the project.

6 To remove a copybook file from the project, highlight the file name in the **Select Files** container and click **Remove**.

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

Figure 3 Cobol Copybook Wizard—Configure Converter Options



New Wizard - Cobol Copybook

Steps

1. Select Wizard Type
2. Select Cobol Copybook Files
3. **Configure Converter Options**

Configure Converter Options

[✓] Ignore copybook content beyond column 72

The converter expects copy books to be width-compliant with IBM's Cobol reference format. Uncheck this box to process copy books with content (excluding comments/line numbers) beyond column 72.

[✓] 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 reserved words for item names. NOTE: When name checking is disabled, the converter cannot process copy books with unnamed items (i.e., implicit 'FILLER' items).

8 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. NOTE: When name checking is disabled, the converter cannot process copy books with unnamed items (i.e., implicit 'FILLER' items). Default: enabled (box is checked).

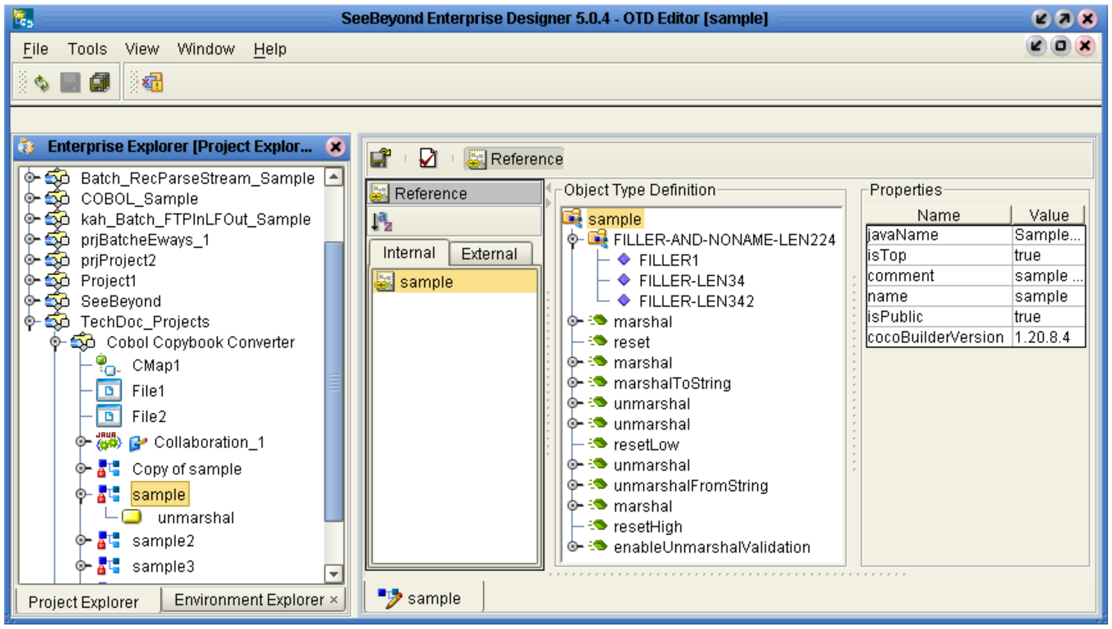
9 Click **Finish**. The **OTD Editor** window appears, displaying the OTD.

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

4.3 Cobol Copybook OTD

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

Figure 4 Sample Copybook OTD



Name	Value
javaName	Sample...
isTop	true
comment	sample ...
name	sample
isPublic	true
cocoBuilderVersion	1.20.8.4

The figure above shows the copybook converter OTD. The OTD has a node for each of the business process 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.

4.4 Cobol Copybook OTD Methods

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

- "Root-level Methods" on page 15
- "Non-Root Methods" on page 19

4.4.1. Root-level Methods

The following methods are the root-level methods provided:

- **"enableUnmarshalValidation(boolean enable)"**
- **"marshal()"**
- **"marshal(String charset)"**
- **"marshal(OtdOutputStream out)"**
- **"marshalToString()"**
- **"reset()"**
- **"resetHigh()"**

- **"resetLow()"**
- **"unmarshal(byte[] in)"**
- **"unmarshal(OtdInputStream in)"**
- **"unmarshal(byte[] in, String charset)"**
- **"unmarshalFromString(String in)"**

enableUnmarshalValidation(boolean enable)

Causes the OTD to validate data flow during an unmarshal call.

Syntax

void enableUnmarshalValidation(boolean enable)

Throws

none.

Examples

TBD

marshal()

Serializes the OTDs content (encoded EBCDIC) as an array of bytes.

Syntax

byte [] marshal()

Throws

MarshalException, IOException

Examples

TBD

marshal(String charset)

This method serializes the content of the OTD as an array of bytes. The content is encoded using the user-specified character set. This method must only be used with copybook OTDs built from copybooks comprised solely of character data records (entries implicitly or explicitly specified as USAGE DISPLAY). Using this method on OTDs designed to hold binary data (e.g., packed decimal, internal decimal) may invalidate the data, because attempts to map portions of the binary content to the specified encoding's character code table may fail. If the specified *charset* value does not name a supported character set, a java.io.UnsupportedEncodingException is generated.

Syntax

byte[] marshal(String charset)

Throws

MarshalException, IOException

Examples

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)

This method serializes the content of the OTD and writes it to the supplied output stream object. The output is EBCDIC data. java.io.IOException is thrown if an output error occurs in attempting to write data to the stream object. A MarshalException occurs if the supplied stream object does not use EBCDIC encoding, or some facet of the OTD content prevents correct serialization; for example, an OTD content such as a binary item may possess a value that exceeds the storage capacity of 8 bytes specified for binary items.

Syntax

void marshal(OtdOutputStream out)

Throws

MarshalException, IOException

marshalToString()

This method serializes the content of the OTD to a String object. This method must only be used 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.

Syntax

String marshalToString()

Throws

MarshalException, IOException

reset()

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

Syntax

void reset()

Throws

none

resetHigh()

Initializes the entire storage space of the OTD to high bit values; each byte is initialized to 0xFF.

Syntax

void resetHigh()

Throws

none

resetLow()

Initializes the OTD storage space to low bit values; each byte is initialized to 0x0.

Syntax

void resetLow()

Throws

none

unmarshal(byte[] in)

Deserializes the given input into an internal data tree.

Syntax

void unmarshal(byte[] in)

Throws

UnmarshalException, IOException

unmarshal(OtdInputStream in)

This method populates the OTD using the supplied OtdInputStream object as the data source. The supplied object must be an opened stream with available data. com.stc.otd.runtime.UnmarshalException is thrown if the data obtained from the stream is incompatible with the OTD, and java.io.IOException is thrown if any other input error occurs in attempting to read data from the stream object. The stream object must flow EBCDIC data (that is, bytes defined in the EBCDIC set).

Syntax

void unmarshal(OtdInputStream in)

Throws

UnmarshalException, IOException

unmarshal(byte[] in, String charset)

This method populates the OTD using the data supplied in the byte array in.

The charset argument specifies the encoding of the given data. If the specified charset value does not name a supported character set, a java.io.UnsupportedEncodingException is generated.

Syntax

void unmarshal(byte[] in, String charset)

Throws

UnmarshalException, IOException

Examples

```
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)

This method populates the OTD using the specified String object as the input source. Generally, this method is useful only to unmarshal character data to copybook OTDs built from copybooks that are comprised solely of character-data records (entries specified implicitly or explicitly as USAGE DISPLAY). Cobol Copybook OTD internally stores content as EBCDIC data, therefore if the String contains characters that cannot be mapped to EBCDIC, the composition of the copied data in the OTD is undefined. For greater control of charset conversion, consider using the unmarshal methods instead.

Syntax

void unmarshalFromString(String in)

Throws

UnmarshalException, IOException

4.4.2. 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*)

Usage Types	**Display**	**COMP or COMP-4**	**COMP-1**	**COMP-2**	**COMP-3**	**COMP-5**	**INDEX**
Data Types							
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						
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	BigDecimal			BigDecimal	BigDeci mal	

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.

Chapter 5

Locating, Importing, and Using the 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 sample that comes with the Cobol Copybook Converter.

In This Chapter

5.1 About the Sample Projects

The Cobol Copybook Converter utility includes the following sample Projects that you can import. This enables you to see how ICAN Projects can work with Cobol copybooks.

- Cobol_BPEL_Sample for use with eInsight/eGate
- Cobol_Copybook_Sample for use with eGate

Each Project contains the following:

- Input data
- Connectivity Maps
- Collaborations
- Business Processes

Version Support

Consult the *Readme.txt* file provided with the sample projects for specific ESR requirements, should they exist, to import or run each of the sample projects.

5.1.1. Cobol_BPEL_Sample

The Cobol BPEL Sample is for use with eInsight. It provides an implementation of the Cobol Copybook Converter that uses the newly supported BPEL functionality. The unzipped Cobol_BPEL_Sample.zip sample project consist of the following files:

- *Cobol_BPEL_Sample.zip* - the project that needs to be imported to an eGate/eInsight installation.
- *qan3glr1.cobol* - the cobol copybook file used for the conversion to create the OTD.
- *inputcobolBPEL.txt* - the input file that the sample project requires when it is run
- *CobolBPELoutput1.dat* - contains the expected output when the project is executed with the given input file

5.1.2. Cobol_Copybook_Sample

The Cobol Copybook Sample is designed for use with eGate. It provides an implementation of the Cobol Copybook Converter that uses Java Collaborations to execute the desired business logic. The unzipped Cobol_Copybook_Sample.zip sample project consist of the following files:

- *EBCDICtoASCII_Sample.zip* - the project that needs to be imported to an eGate installation.
- *misco1a.cobol* - the cobol copybook file used for the conversion to create the OTD.
- *input.txt* - the input file that the sample project requires when it is run.
- *COBOLoutput1.dat* - contains 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 forward the output to an outbound File eWay. The resulting file output is the ASCII translation of the original input data.

5.2 Locating the Sample Projects

The sample Projects are included in the **CobolCopyBookDocs.sar**. This file is uploaded separately from the Cobol Copybook sar file during installation. For information, refer to **"Installing the Cobol Copybook Converter" on page 10**.

Once you have uploaded the **CobolCopyBookDocs.sar** to the Repository and you have downloaded the sample Projects (**Cobol_Copybook_Sample.zip**) using the **DOCUMENTATION** tab in the Enterprise Manager, the sample resides in the folder specified during the download.

5.3 Importing the Sample Projects

This section describes the process required to import each of the sample projects into the Enterprise Designer.

To import the eGate sample

1. Unzip the **Cobol_Converter_Sample.zip** file to a temporary directory.

   For information about locating this file, refer to **"Locating the Sample Projects" on page 22**.

2. In the Project Explorer tab of the Enterprise Designer, right-click the Repository and click **Import Project**. The **Select File to Import** dialog box appears.

3. Browse to the temporary directory.

4. Double-click **EBCDICtoASCII_Sample.zip.** The **File Destination** dialog box appears.

5. Click **Import to a new Project**, enter the name of the Project, and click **OK**.

6. When the import has successfully completed, right-click the Repository and click **Refresh All from Repository**.

The Project is now imported. Before you deploy and run the Project, do the following:

- Configure the eWays for the correct input and output directories. Refer to the eWay documentation for more information.
- Create an Environment and Deployment Profile, and run the Project. Refer to the *eGate Integrator User's Guide* for more information.

To import the eInsight sample

1. Unzip the **Cobol_Converter_Sample.zip** file to a temporary directory.

   For information about locating this file, refer to **"Locating the Sample Projects" on page 22**.

2. In the Project Explorer tab of the Enterprise Designer, right-click the Repository and click **Import Project**. The **Select File to Import** dialog box appears.

3. Browse to the temporary directory.

4. Double-click **Cobol_BPEL_Sample.zip.** The **File Destination** dialog box appears.

5. Click **Import to a new Project**, enter the name of the Project, and click **OK**.

6. When the import has successfully completed, right-click the Repository and click **Refresh All from Repository**.

The Project is now imported. Before you deploy and run the Project, do the following:

- Configure the eWays for the correct input and output directories. Refer to the eWay documentation for more information.
- Create an Environment and Deployment Profile, and run the Project. Refer to the *eGate Integrator User's Guide* for more information.
-

5.4 Running the Sample Projects

The sample Projects do not include the eGate Environments, Deployment Profiles, and the physical configurations for the eWays needed to deploy the Projects. The steps required to run the sample projects include:

1. Create an Environment Profile as described in the *eGate Integrator User's Guide.*
2. Create a Deployment Profile as described in the *eGate Integrator User's Guide.*
3. Run the Project as described in the *eGate Integrator User's Guide.*

5.5 Building Cobol Copybook Business Logic with eInsight

This section describes how to build the business logic with eInsight:

- **Adding a New Business Process** on page 24
- **Building the Business Processes** on page 24
- **Creating the Connectivity Map** on page 26
- **Binding the Business Process and eWays** on page 27

To see an example of Business Processes and Connectivity Maps, import the Cobol_BPEL_Sample sample Project as described in **"Locating, Importing, and Using the Sample Projects" on page 21**.

5.5.1. Adding a New Business Process

To add Business Processes

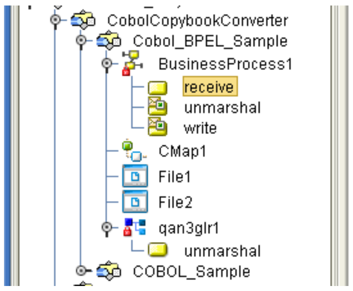
- In the **Project Explorer** tab of the Enterprise Designer, right-click the Project for which you intend to create a Business Process, click **New**, and then **Business Process**.

5.5.2. Building the Business Processes

To build Business Processes

1. In the **Project Explorer** tab of the Enterprise Designer, expand the OTD. This displays the OTD methods.

Figure 5 Cobol Copybook OTD Methods



2 Drag the *unmarshal* Cobol Copybook OTD method to the Business Process Designer canvas.

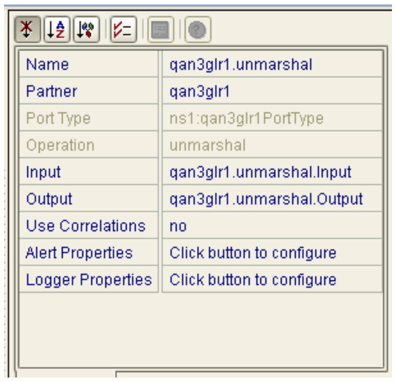
3 Expand the **SeeBeyond**, **eWays**, **File**, and **FileClient** folders in the **Project Explorer** tab.

4 Drag the *write* method to the Business Process Designer canvas.

5 Drag the *receive* method to the Business Process Designer canvas.

6 Click the *unmarshal* Business Activity and click **Show Properties**. The **Properties** dialog box appears.

Figure 6 UnMarshal Properties



Name	qan3glr1.unmarshal
Partner	qan3glr1
Port Type	ns1:qan3glr1PortType
Operation	unmarshal
Input	qan3glr1.unmarshal.Input
Output	qan3glr1.unmarshal.Output
Use Correlations	no
Alert Properties	Click button to configure
Logger Properties	Click button to configure

7 Click the **Input** box and select **%OTDNAME%.unmarshal.input**.

8 Click the **Output** box and select **%OTDNAME%.unmarshal.output**.

9 Configure all other Activities by highlighting the Activity and clicking **Show Properties**. Refer to **“Cobol Copybook OTD Methods” on page 15** for Business Operations syntax.

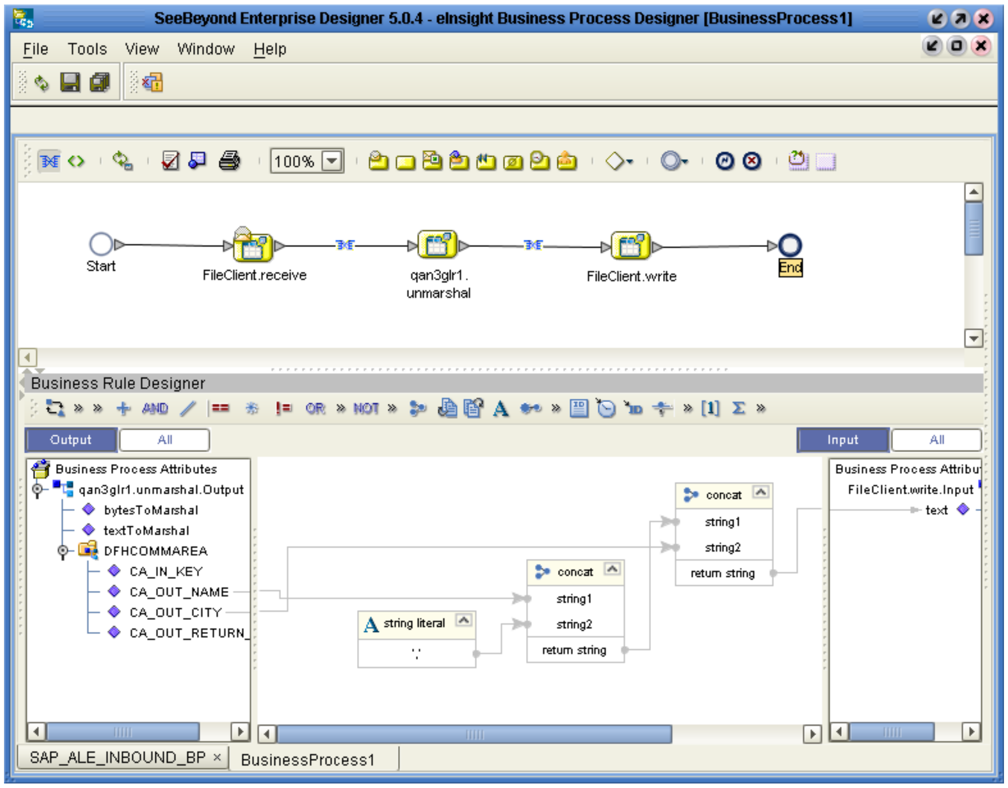
10 Link all components as described in *eInsight Business Process Manager User’s Guide*.

11 To create data mappings, right-click the link between the Activities and click **Add Business Rule**.

12 In the **Business Rule Editor** window, create the code and the data mappings. For details, refer to the *eInsight Business Process Manager User’s Guide*.

Figure 7 on page 26 shows an example of an Business Process including the data mapping in the **Business Rule Editor** window. To explore the business logic design for an actual Project, import the Cobol_BPEL_Sample sample Project as described in **"Importing the Sample Projects" on page 23**.

Figure 7 Business Process and Data Mapping



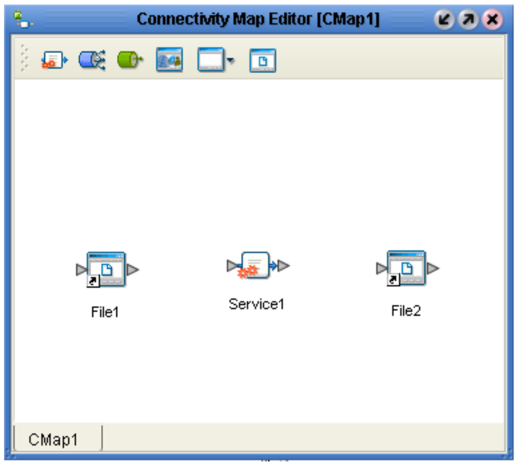
5.5.3. Creating the Connectivity Map

The procedure below describes how to create the Connectivity Map for the COBOL copybook conversion Project.

To create the Connectivity Map

1. In the Project Explorer tab of the Enterprise Designer, right-click the copybook conversion Project, click **New**, and click **Connectivity Map**. A blank Connectivity Map appears.
2. Click the **eWay** icon and click the eWay type.
3. Drag the eWay icon to the Connectivity Map to create the inbound eWay.
4. Drag the **Service** icon to the Connectivity Map.
5. Click the eWay icon and click the eWay type.
6. Drag the eWay icon to the Connectivity Map to create the outbound eWay. The Connectivity Map looks similar to the figure below.

Figure 8 COBOL Copybook Conversion Connectivity Map



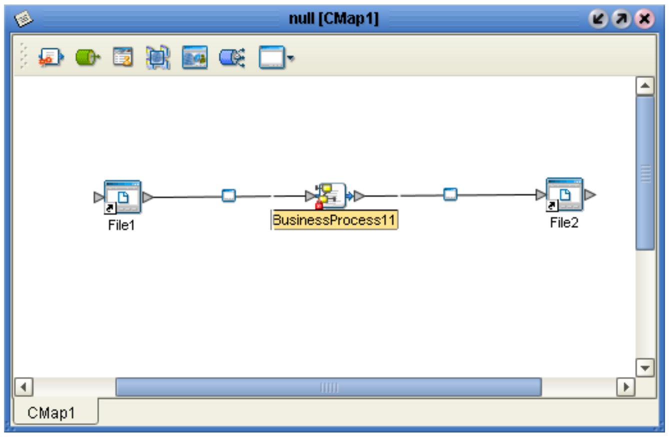
5.5.4. Binding the Business Process and eWays

Once you have created the business process and its business logic, you can bind the new business process to the Service, and then connect the business process to the eWays.

To bind the Business Process and eWays

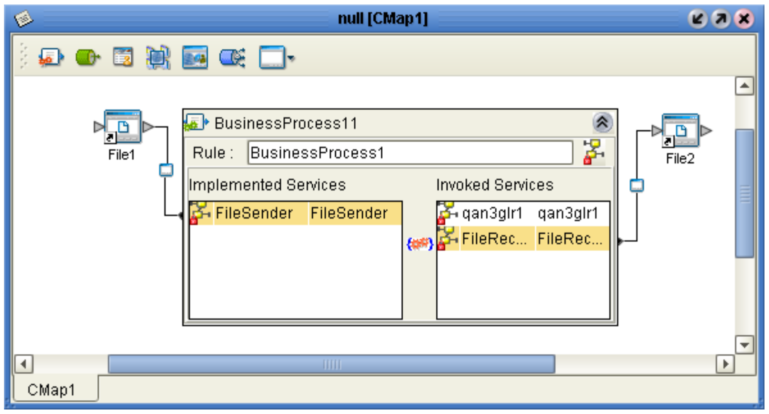
1. From the Project Explorer of the Enterprise Designer, drag the newly created Business Process to the Service in the Connectivity Map as shown below.

Figure 9 Binding the Business Process and Service



2. Double-click the **Service** icon. The **Service1** window appears.
3. Drag the input service to the inbound eWay. For example, for a File eWay, the input service is **FileSender**.
4. Drag the output service to the outbound eWay as shown below.

Figure 10 Connecting the Business Process to the eWays



5 Close the **BusinessProcess11** window and click **Save**.

Once you have completed the Connectivity Map binding, you must do the following to finish the Project:

1 Configure the File eWays as described in the eWay documentation.

2 Create an Environment and Deployment Profile and run the Project as described in the *eGate Integrator User's Guide.*

5.6 Building Cobol Copybook Business Logic with eGate

This section describes how to use the Cobol Cobybook Converter to convert files using COBOL copybooks. As a quick start, the following list provide an overview of the steps taken:

1 Create an eGate Project if necessary.

2 Create a Cobol Object Type Definition (OTD) that indicates to the Collaboration to receive data, use the supplied COBOL copybook file to convert it, and forward the converted data to an output eWay - see **"Creating a COBOL Copybook Project and OTD" on page 29**.

3 Create a Connectivity Map with an inbound eWay, a Collaboration, and an outbound eWay - see **"Creating the Connectivity Map" on page 29**.

4 Creating the Collaboration Definition and its business logic - see **"Creating the Collaboration Definition" on page 30** and **"Building Collaboration Definitions" on page 32**.

5 Bind the newly created Cobol OTD to the Collaboration and connect the Collaboration to the eWays - see**"Binding the Collaboration Definition and eWays" on page 37**.

6 Create an eGate Environment - see **"Running the Sample Projects" on page 24**.

7 Create a Deployment Profile - see **"Running the Sample Projects" on page 24**.

8 Deploy and run the Project - see **"Running the Sample Projects" on page 24**.

To see an example of Cobol Copybook Converter Collaborations and Connectivity Maps, import the EBCDICtoASCII sample Project as described in **"Locating, Importing, and Using the Sample Projects" on page 21**.

5.6.1. Creating a COBOL Copybook Project and OTD

See "Creating Cobol Copybook OTDs" on page 12 for details about how to create a Cobol Copybook converter OTD.

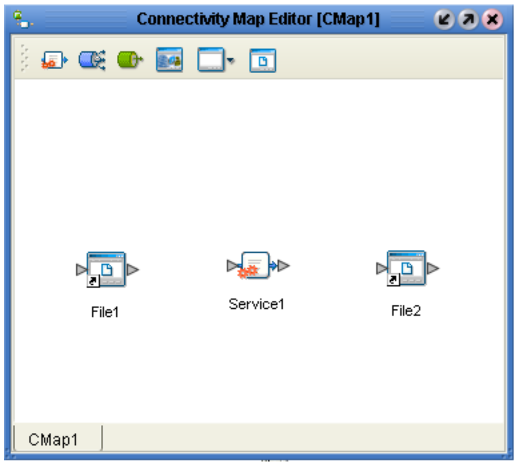
5.6.2. Creating the Connectivity Map

The procedure below describes how to create the Connectivity Map for the COBOL copybook conversion Project.

To create the Connectivity Map

1. In the Project Explorer tab of the Enterprise Designer, right-click the copybook conversion Project, click **New**, and click **Connectivity Map**. A blank Connectivity Map appears.
2. Click the **eWay** icon and click the eWay type.
3. Drag the eWay icon to the Connectivity Map to create the inbound eWay.
4. Drag the **Service** icon to the Connectivity Map.
5. Click the eWay icon and click the eWay type.
6. Drag the eWay icon to the Connectivity Map to create the outbound eWay. The Connectivity Map looks similar to the figure below.

Figure 11 COBOL Copybook Conversion Connectivity Map



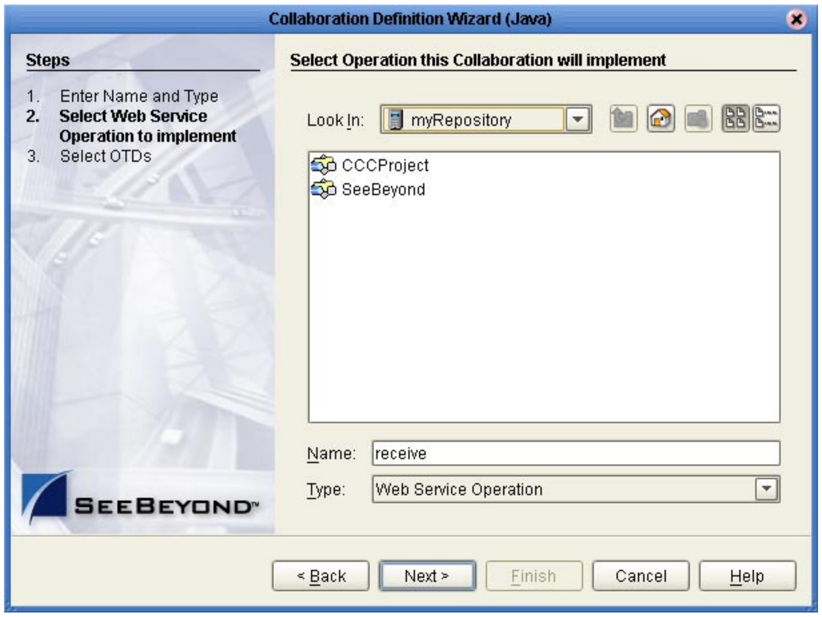
5.6.3. Creating the Collaboration Definition

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 12 Selecting Collaboration Operations


Collaboration Definition Wizard (Java)
Steps
1. Enter Name and Type
2. Select Web Service Operation to implement
3. Select OTDs
SEEBEYOND
< Back
Next >
Finish
Cancel
Help

Select Operation this Collaboration will implement

Look In: myRepository

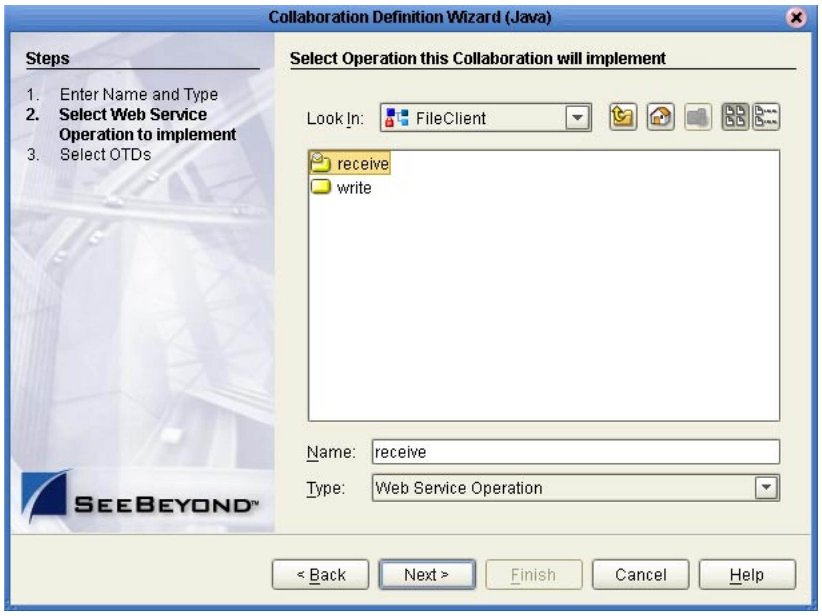
CCCProject
SeeBeyond

Name: receive

Type: Web Service Operation

3 Double-click **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 13 Selecting File Receive


Collaboration Definition Wizard (Java)
Steps
1. Enter Name and Type
2. Select Web Service Operation to implement
3. Select OTDs
SEEBEYOND
< Back
Next >
Finish
Cancel
Help

Select Operation this Collaboration will implement

Look In: FileClient

receive
write

Name: receive

Type: Web Service Operation

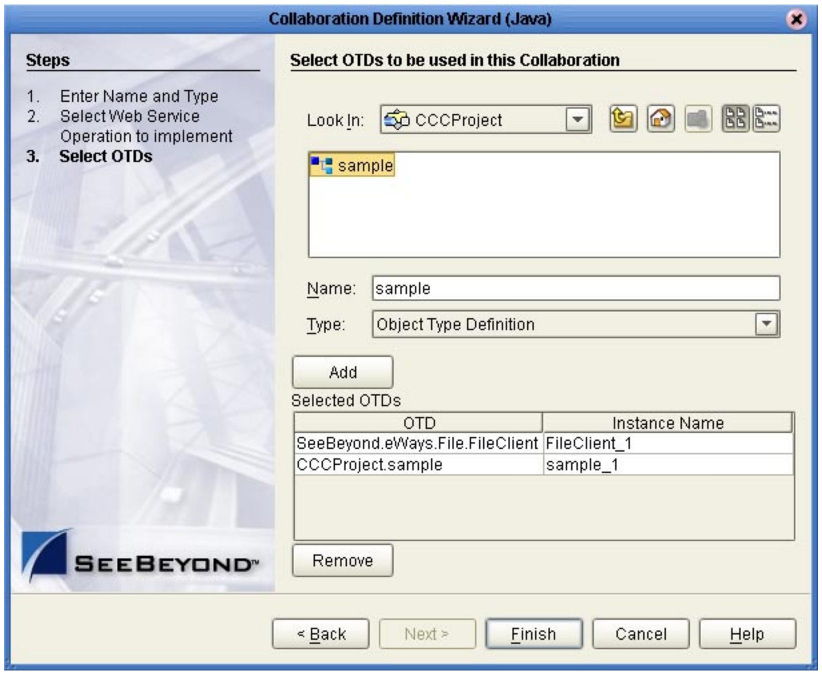
4 Click **Next**.

5 Double-click **SeeBeyond**, **eWays—**continue to double-click to select the outbound eWay and the (outbound) web service. For example, for the a 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 14 Completed Collaboration Definition



Select OTDs to be used in this Collaboration

Look In: CCCProject

sample

Name: sample

Type: Object Type Definition

Selected OTDs

OTD	Instance Name
SeeBeyond.eWays.File.FileClient	FileClient_1
CCCProject.sample	sample_1

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

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

5.6.4. Building Collaboration Definitions

Once you have created the Collaboration Definition as described in the section above, you can create the business logic for the Collaboration. The business logic for a copybook conversion consist of the following components;

1 **Unmarshaling the Input Formats** on page 32

2 **Specifying Destinations** on page 35

3 **Writing The Output to a File** on page 37

Unmarshaling the Input Formats

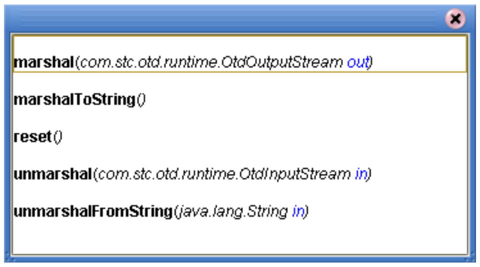
The first step in the business logic is to handle the data when it comes into the Project. The Cobol Copybook OTD can process text data, and as such, text data can easily be unmarshaled with the **unmarshalFromString method()**.

For other data, you must convert the array data into an array input stream, and then into an OTD input stream.

To unmarshal text input format

1 Right-click the copybook OTD and click **Select a method to call**. A list of methods appears.

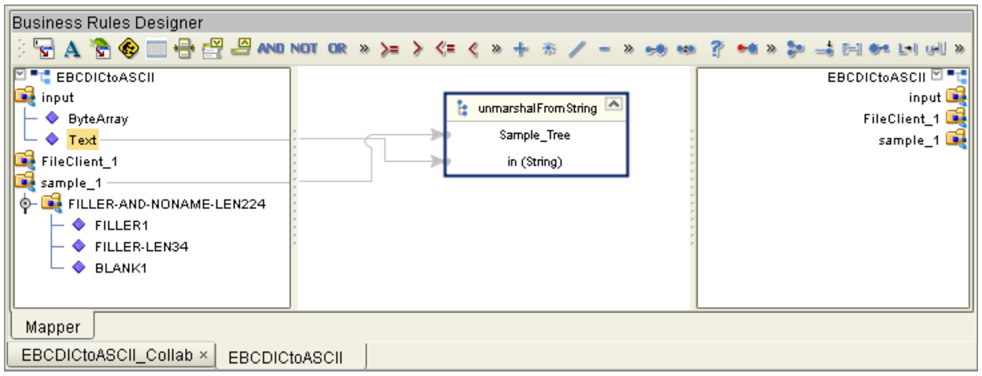
Figure 15 Cobol Copybook Converter Methods


marshal(*com.stc.otd.runtime.OtdOutputStream out*)
marshalToString()
reset()
unmarshal(*com.stc.otd.runtime.OtdInputStream in*)
unmarshalFromString(*java.lang.String in*)

2 Click **unmarshalFromString()**. The **unmarshalFromString** box appears.

3 Expand the input node and drag **Text** into in **(String)** as shown below.

Figure 16 Unmarshaling Text Input



To handle bytes input format

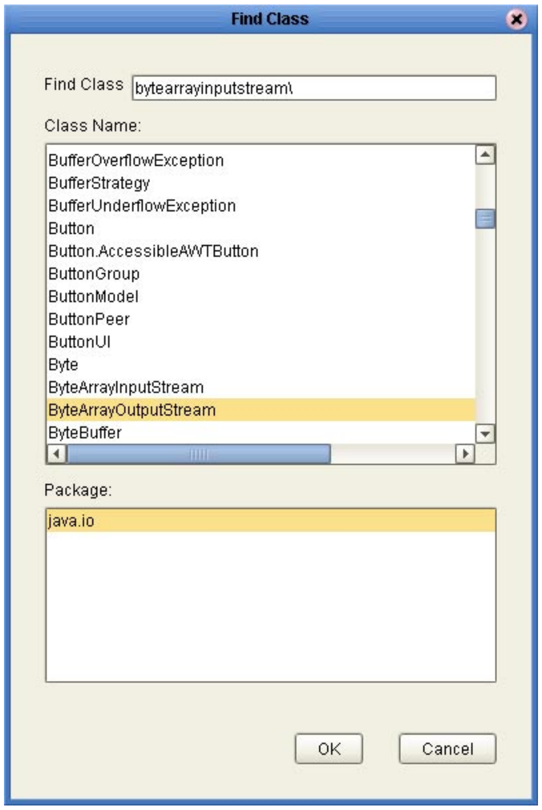
1 Click **Local Variable**. The **Create a variable** dialog box appears.

2 In the **Variable Name** box, enter the variable name.

3 Click **Class** and the ellipsis button. The **Find Class** dialog box appears.

4 In the **Find Class** box, type **bytearray** and press **ENTER**. The **Find Class** dialog box shows the package available for the ByteArrayInputStream as shown below.

Figure 17 Creating a ByteArrayInputStream Variable



5 Click **OK** twice.

6 Click **Local Variable** to create the second variable to convert the array input stream to the OTD input stream. The **Create a variable** dialog box appears.

7 In the **Variable Name** box, enter the name of the variable, for example, OTDstream.

8 In the **Class** box, type:

com.stc.otd.runtime.OtdInputStream

9 Click **OK**. This add the following business rule:

com.stc.otd.runtime.OtdInputStream.otdstream;

10 Click **Source code mode** and scroll to the business rule.

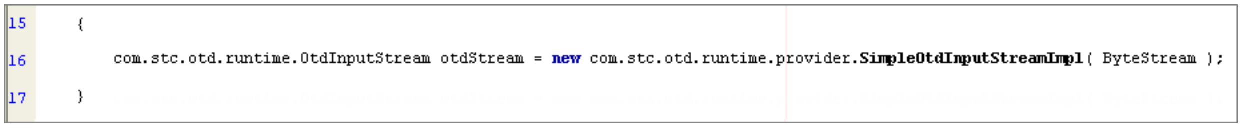
11 Delete the semi colon at the end of the line.

12 Add the following code:

= new com.stc.otd.runtime.provider.SimpleOtdInputStreamImpl(***firstvariable*** **);**

Where *firstvariable* is the variable created in step 6.

Figure 18 Added Variable Code



```
    {
        com.stc.otd.runtime.OtdInputStream otdStream = new com.stc.otd.runtime.provider.SimpleOtdInputStreamImpl( ByteStream );
    }
```

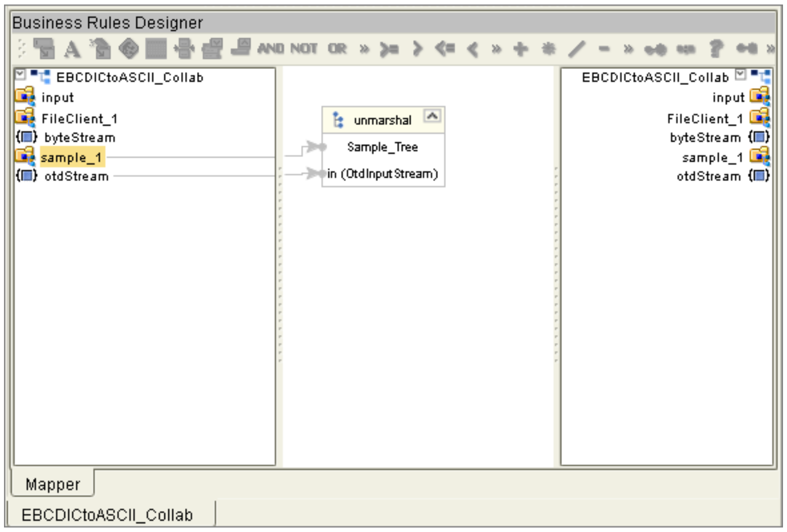
13 Click **Commit Changes**.

14 Right-click the copybook OTD (with the copybook filename) and click **Select a method to call**.

15 Click **unmarshal()**. This adds the **unmarshal** box.

16 Drag the *firstvariable* created in step 6 to **in (OtdInputStream)** as shown below.

Figure 19 Unmarshaling Non-String Data



Specifying Destinations

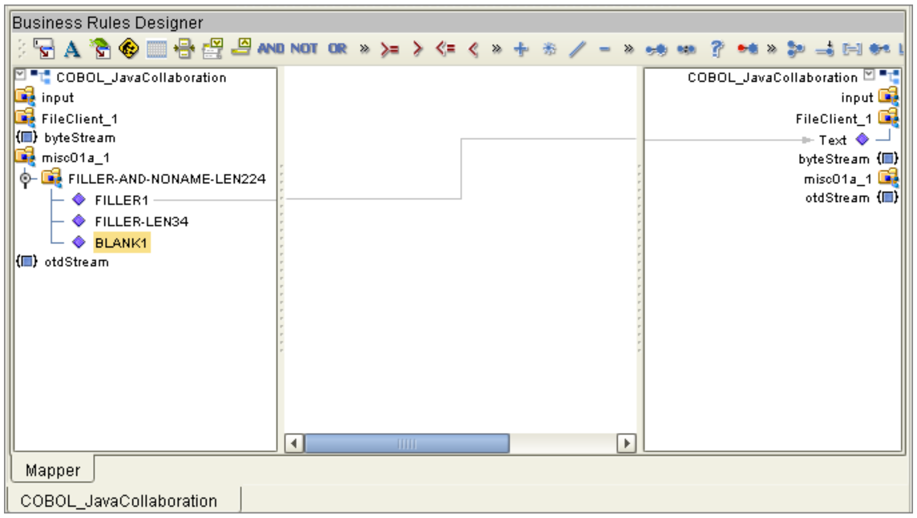
You can specify destinations by mapping specific input data to output data, or you can marshal the data to the destination.

To map input and output data

1 Expand the input OTD node.

2 Drag the input nodes to the output data type under the output service as shown below.

Figure 20 Mapping Input and Output Data

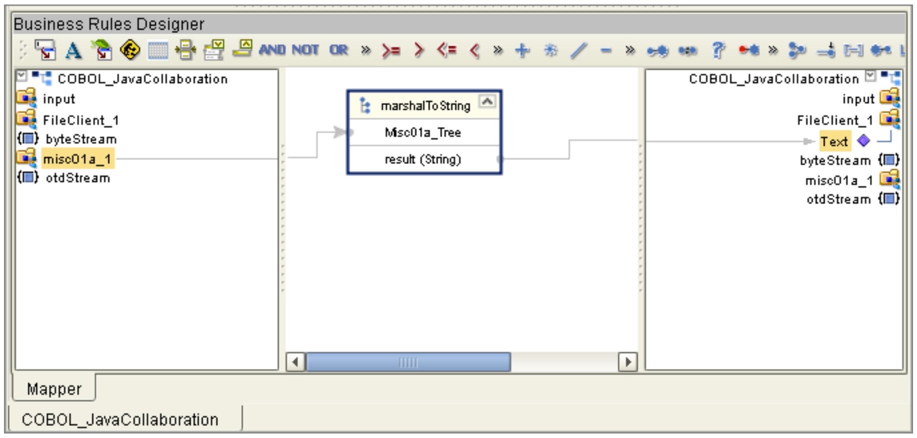


To marshal data as strings to an output destination

1 Right-click the copybook OTD, click **Select a method to call**, and click **marshalToString()**. The **marshalToString** box appears.

2 Drag **Result (String)** to the output OTD as shown below.

Figure 21 Marshaling Data as String to an Output Destination



To marshal data to an output destination

1 Right-click the copybook OTD, click **Select a method to call**, and click **marshal()**. The **marshal** box appears.

2 Drag **Out (OtdOutputStream)** to the appropriate payload node of the output OTD. Verify that the marshal method you selected has a result type compatible with the payload type.

Writing The Output to a File

If you are using a File eWay for the output of the copybook conversion Project, you can use the method below to write the output to a file.

To write the output to a file

1. Right-click **FileClient_1** in the input column, click **Select a method to call**, and click **write()**.
2. Click **Save**.

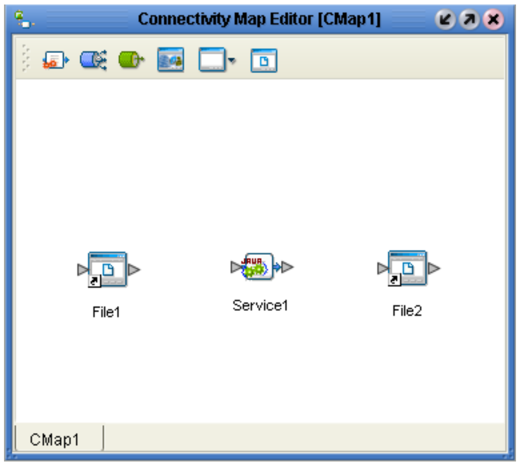
5.6.5. Binding the Collaboration Definition and eWays

Once you have created the Collaboration and its business logic as described in the section above, you can bind the new Collaboration Definition to the Service, and then connect the Collaboration to the eWays.

To bind the Collaboration Definition and eWays

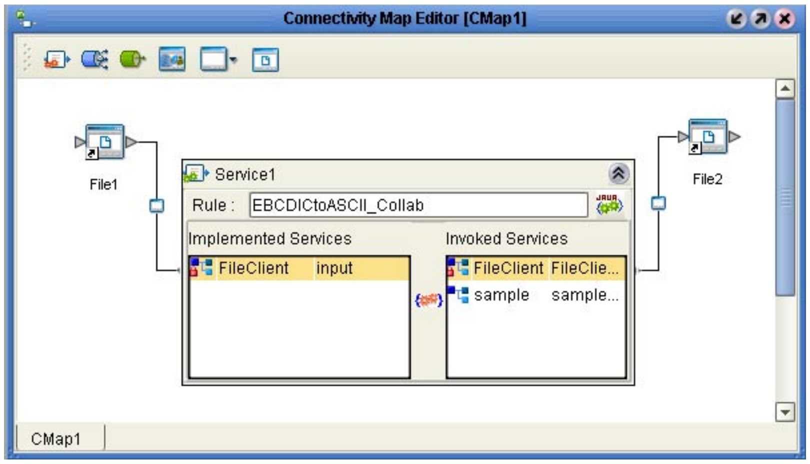
1. From the Project Explorer of the Enterprise Designer, drag the newly created Collaboration Definition to the Service in the Connectivity Map as shown below.

Figure 22 Binding the Collaboration Definition and Service



2. Double-click the **Service** icon. The **Service1** window appears.
3. Drag the input service to the inbound eWay. For example, for a File eWay, the input service is **FileClient input**.
4. Drag the output service to the outbound eWay as shown below.

Figure 23 Connecting the Collaboration to the eWays



5 Close the **Service1** window and click **Save**.

Once you have completed the Connectivity Map binding, you must do the following to finish the Project:

1 Configure the File eWays as described in the eWay documentation.

2 Create an Environment and Deployment Profile and run the Project as described in the *eGate Integrator User's Guide.*

Index