Cobol Copybook Converter
User’s Guide

Release 5.0.3
Contents

Chapter 1

Introduction 5

Contents of This Guide 5

Writing Conventions 5

Additional Conventions 6

Supporting Documents 7

SeeBeyond Web Site 7

Chapter 2

About the Cobol Copybook Converter 8

Introducing the Cobol Copybook Converter 8

Supported Operating Systems 9

System Requirements 9

Chapter 3

Installing the Cobol Copybook Converter 10

Installing the Cobol Copybook Converter 10

Chapter 4

Using the Cobol Copybook Converter 11

Converting Files with COBOL Copybooks 11

Creating a COBOL Copybook Project and OTD 12

Creating the Connectivity Map 13

Creating the Collaboration Definition 14

Creating the Business Logic for The Collaboration Definition 16

Unmarshaling the Input Formats 16

Specifying Destinations 19

Writing The Output to a File 21

Binding the Collaboration Definition and eWays 21

Cobol Copybook Converter Methods 22
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Root-level Methods</strong></td>
<td>22</td>
</tr>
<tr>
<td>byte[] marshal(String charset)</td>
<td>22</td>
</tr>
<tr>
<td>void unmarshal(byte[] in, String charset)</td>
<td>23</td>
</tr>
<tr>
<td>void unmarshal(OtdInputStream)</td>
<td>23</td>
</tr>
<tr>
<td>void unmarshalFromString(String)</td>
<td>24</td>
</tr>
<tr>
<td>void marshal(OtdOutputStream)</td>
<td>24</td>
</tr>
<tr>
<td>String marshalToString</td>
<td>24</td>
</tr>
<tr>
<td><strong>Non-Root Methods</strong></td>
<td>24</td>
</tr>
<tr>
<td><strong>Working with the Cobol Copybook Converter Sample</strong></td>
<td>26</td>
</tr>
<tr>
<td>Locating the Sample Project</td>
<td>26</td>
</tr>
<tr>
<td>About the Sample Project</td>
<td>26</td>
</tr>
<tr>
<td>Importing the Sample Project</td>
<td>26</td>
</tr>
<tr>
<td><strong>Index</strong></td>
<td>28</td>
</tr>
</tbody>
</table>
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, “About 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 4, “Using the Cobol Copybook Converter” on page 11 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

<table>
<thead>
<tr>
<th>Text</th>
<th>Convention</th>
<th>Example</th>
</tr>
</thead>
</table>
| 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](http://www.seebeyond.com) |

Additional Conventions

**Windows Systems**

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

**Path Name Separator**

This guide uses the backslash (“\”) as the separator within path names. If you are working on a UNIX system, please make the appropriate substitutions.
1.3 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

About the Cobol Copybook Converter

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

In This Chapter

- Introducing the Cobol Copybook Converter on page 8
- Supported Operating Systems on page 9
- System Requirements on page 9

2.1 Introducing 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; 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 presents the copybook specification to the Cobol Copybook Converter in 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**

Copybooks per Cobol standard do not contain content beyond column 72. However, if the copybook to be converted does contain content beyond column 72, you can configure the Copybook Converter to convert the content. To do so, you deselect the the **Ignore copybook content beyond column 72** option as described on page 13. Use this option with caution; the copybook’s compliance with COBOL copybook standards determines whether the copybook processes correctly. The example below shows an example of a copybook entry that exceeds 72 columns, but will pass:

```
1234567890123456789012345678901234567890123456789012345678901234567890
 060001 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).

### 2.2 Supported Operating Systems

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

- Windows 2000 SP3

### 2.3 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*.

The system were Cobol Copybook Converter is installed needs approximately 20 MB of free disk space for the application and its configuration, library, and script files.
Chapter 3

Installing the Cobol Copybook Converter

This chapter describes how to install the Cobol Copybook Converter.

In This Chapter

- “Installing the Cobol Copybook Converter” on page 10

3.1 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 Project)
   - CobolCopyBookDocs.sar (to install the user guide and the sample Project)

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 “Working with the Cobol Copybook Converter Sample” on page 26.
Chapter 4

Using the Cobol Copybook Converter

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

- Converting Files with COBOL Copybooks on page 11
- Cobol Copybook Converter Methods on page 22
- Working with the Cobol Copybook Converter Sample on page 26

4.1 Converting Files with COBOL Copybooks

This section describes how to use the Cobol Copybook 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.
3. Create a Connectivity Map with an inbound eWay, a Collaboration, and an outbound eWay.
5. Bind the newly created Cobol OTD to the Collaboration and connect the Collaboration to the eWays.
6. Configure the eWays if necessary.
7. Create an eGate Environment and Deployment Profile.
8. Deploy and run the Project.
4.1.1. Creating a COBOL Copybook Project and OTD

To create a COBOL copybook Project and OTD

1. In the Project Explorer tab of the Enterprise Designer, right-click the Repository and click New Project.

2. Right-click the new Project, click New, and click Object Type Definition. The New Object Type Definition wizard appears as shown below.

![New Object Type Definition Wizard](image)

3. Click Cobol Copybook and click Next. The Select Cobol Copybook Files page appears.
4.1.2. 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. In the **Look In** box, browse to the location where the COBOL copybook file resides which contains the specifications for this conversion.

5. Deselect the **Ignore copybook content beyond column 72** option only if your copybook area B includes information after column 72 that needs to be processed. **Note:** Use this option with caution; the copybook's compliance with COBOL copybook standards determines whether the copybook processes correctly. For more information, refer to page page 9.

6. Double-click the file and click **Finish**. The **OTD Editor** window displays the newly created OTD. For information about Cobol Copybook Converter methods, refer to “Cobol Copybook Converter Methods” on page 22.
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 3** COBOL Copybook Conversion Connectivity Map

![Connectivity Map](image)

4.1.3. **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.
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 5  Selecting File Receive**

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 6** Completed Collaboration Definition

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

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

### 4.1.4. Creating the Business Logic for The Collaboration Definition

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 16

2 **Specifying Destinations** on page 19

3 **Writing The Output to a File** on page 21

#### 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 7** Cobol Copybook Converter Methods

   ```
   marshal(com.stc.dev.runtime.OTDOutputStream.out)
   marshalToStream()
   result()
   unmarshal(com.stc.dev.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 8** 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 9  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:

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

   Where `firstvariable` is the variable created in step 6.
Figure 10  Added Variable Code

```java
15  {
16    com.stor.object.90dInputStream otdStream = new com.stor.object.provider.SimpleOTDInputStream Impl ByteStream();
17  }
```

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 11  Unmarshaling Non-String Data](image)

### 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 12  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 13  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.

4.1.5. 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 14  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.
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 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*.

---

### 4.2 Cobol Copybook Converter Methods

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

#### 4.2.1. Root-level Methods

The following methods are the root-level methods provided.

- `byte[] marshal(String charset)`
- `void unmarshal(byte[] in, String charset)`
- `void unmarshal(OtdInputStream)`
- `void unmarshalFromString(String)`
- `void marshal(OtdOutputStream)`
- `String marshalToString()`

**byte[] 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. If the OTD content contains data that
cannot be mapped to the specified character set, the data returned may be invalid. If the specified charset value does not name a supported character set, a java.io.UnsupportedEncodingException is generated.

**Syntax**

```java
byte[] marshal(String charset)
```

**Throws**

UnmarshalException, IOException

**Examples**

```java
byte[] content = cocoOtd.marshal("cp037"); // retrieve OTD content as EBCDIC data
byte[] content = cocoOtd.marshal("US-ASCII"); // retrieve OTD content as ASCII data
```

**void unmarshal(byte[] in, String charset)**

This method populates the OTD using the data supplied in the byte array `in`. The charset argument specifies the character set to use to decode the content of the input array. If the content contains data that cannot be mapped to the specified character set, the copy of the data stored in the OTD may lose information. If the specified charset value does not name a supported character set, a java.io.UnsupportedEncodingException is generated.

**Syntax**

```java
void unmarshal(byte[] in, String charset)
```

**Throws**

UnmarshalException, IOException

**Examples**

```java
byte[] bytes = ... 
cocoOtd.unmarshal(bytes, "cp037"); // Interpret bytes content as EBCDIC data
cocoOtd.unmarshal(bytes, "US-ASCII"); // Interpret bytes content as ASCII data
```

**void unmarshal(OtdInputStream)**

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**

```java
void unmarshal(OtdInputStream)
```

**Throws**

UnmarshalException, IOException
void unmarshalFromString(String)

This method populates the OTD using the specified String object as the input source. Cobol Copybook OTD internally stores content as EBCDIC data, therefore if the String contains characters 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)

Throws

UnmarshalException, IOException

void marshal(OtdOutputStream)

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)

Throws

MarshalException, IOException

String marshalToString

This method serializes the content of the OTD to a String object.

Syntax

String marshalToString()

Throws

MarshalException, IOException

4.2.2. 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\((T)\)

<table>
<thead>
<tr>
<th>Usage Types</th>
<th>Display</th>
<th>COMP or COMP-4</th>
<th>COMP-1</th>
<th>COMP-2</th>
<th>COMP-3</th>
<th>COMP-5</th>
<th>INDEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Types</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alphabetic</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>For example: PIC AAA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alphanumeric</td>
<td>String</td>
<td>String</td>
<td>String</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>For example: PIC X9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alphanumeric edited</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>For example: PIC X89</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Numeric edited</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>For example: PIC ZZZ99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBCS</td>
<td>byte[]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>For example: PIC GGBGG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>External floating point</td>
<td>BigDecimal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>For example: PIC +9V99E+99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Numeric integer (9 digits or less)</td>
<td>int</td>
<td>int</td>
<td></td>
<td>int</td>
<td>int</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>Numeric floating point (COMP-1 or COMP-2 items)</td>
<td>BigDecimal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Numeric Integer (10 to 18 digits)</td>
<td>long</td>
<td>long</td>
<td></td>
<td>long</td>
<td>long</td>
<td>long</td>
<td></td>
</tr>
<tr>
<td>Numeric integer (19 digits or more)</td>
<td>BigDecimal</td>
<td>BigDecimal</td>
<td>BigDecimal</td>
<td>BigDecimal</td>
<td>BigDecimal</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

### 4.3 Working with the Cobol Copybook Converter Sample

Cobol Copybook Converter includes a sample Project that you can import to see how a COBOL conversion can be set up. This section describes where the sample Project resides and how to import and use it.

#### 4.3.1. Locating the Sample Project

The sample Project is 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 Project (**Cobol_Copybook_Sample.zip**) using the DOCUMENTATION tab in the Enterprise Manager, the sample resides in the folder specified during the download.

#### 4.3.2. About the Sample Project

The sample Project zip file contains:

- A zip file with an exported Project that you can import (**EBCDICtoASCII_Sample.zip**)
- An input file
- A COBOL copybook to be used for the conversion

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.

#### 4.3.3. Importing the Sample Project

To import the Project into Enterprise Designer, follow the steps below.

**To import the sample**

1. Unzip the **Cobol_Converter_Sample.zip** file to the temporary directory.
   
   For information about locating this file, refer to “Locating the Sample Project” on page 26.

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.
Index

Numerics
72 column area B inclusion 13

B
byte marshal(String charset) 22

C
CICS 8
   COMM AREA 8
Cobol Copy statements 8
Cobol Copybook Converter overview 8
COMM AREA 8
conventions
   path name separator 6
   Windows 6
converter methods 22
converting files 11

D
document
   conventions 5
document purpose and scope 5

I
implementation 11
installing 10
introduction 8

J
Java methods 22

M
methods 22

O
organization of information, document 5
overview, Cobol Copybook Converter 8

P
PIC X(4) 9
platforms, supported 9

R
requirements 9

S
SeeBeyond Web site 7
   statements
      Cobol Copy 8
      usage pointer 9
String marshalToString 24
supported platforms 9
supporting documents 7
system requirements 9

T
technical support
   SeeBeyond Web site 7

U
unsupported features 8
usage pointer statements 9
using Cobol Copybook Converter 11

V
void marshal(OtdOutputStream) 24
void unmarshal(byte in, String charset) 23
void unmarshal(OtdInputStream) 23
void unmarshalFromString(String) 24

W
Windows 2000 8
Windows NT 8
writing conventions 5