Oracle® Java CAPS COBOL Copybook Encoder User's Guide

Part No: 821–2637
March 2011

License Restrictions Warranty/Consequential Damages Disclaimer

Warranty Disclaimer

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

Restricted Rights Notice

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

Hazardous Applications Notice

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Trademark Notice

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group in the United States and other countries.

Third Party Content, Products, and Services Disclaimer

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

120126@25097

Contents

Working With the COBOL Copybook Encoder

This document covers the following topics:

COBOL Copybook Encoder Overview

The COBOL Copybook Encoder converts copybook descriptions, and creates schema definitions designed to encapsulate data conforming to the description. The encoder can generate the schema definition based on a copybook file.

About COBOL Copybooks

Copybooks are common fragments of code that are typically distributed throughout a software application. They are functionally similar to the #include file of a C or C++ application, and mainframes reference these books and call structures as needed. They are usually stored in a source library file. 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, disparate applications cannot communicate with each other and data cannot be transferred between applications and platforms. Copybook documents use a fixed-length field encoding mechanism.

About the COBOL Copybook Encoder

The COBOL Copybook Encoder allows you to handle data encoded in COBOL copybook format in a JBI application. With the encoder, you can create JBI applications that can read or send data in copybook format, connecting mainframe applications to applications running on other platforms. The COBOL Copybook Encoder definition is represented by an XSD file and is used in the WSDL document to define the encoding. The encoding style is specified in the WSDL document's binding element. The encoding style for COBOL copybook is `cocoencoder-1.0`.

The COBOL Copybook Encoder is part of a larger framework of custom encoders provided with Java CAPS. A standard encoder is a bidirectional software component that transforms an XML message into a non-XML message, or the other way around. In this case, it encodes data from XML to copybook, and from copybook to XML. For more information about custom encoders, see the *Oracle Java CAPS Custom Encoders User's Guide*.

Handling 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 (that is, content that is not line numbering or comments, which should be ignored), you need to specify not to ignore the content when you create the XSD file for the copybook document.

Caution – It is still possible for a copybook with data beyond the 72nd column to process successfully if you choose to ignore the extra content. However, the data may not be process correctly.

Creating the Project Files

To implement COBOL copybook encoding in your JBI projects, you need to import or create the copybook file, create an XSD file (where the encoding is configured), and add the encoding type to the WSDL document. The following topics provide instructions for each of these steps.

- "Creating the COBOL Copybook Document" on page 6
- "Generating the XML Schema Definition (XSD)" on page 10
- "Creating a WSDL Document to Use COBOL Copybook Encoding" on page 12

Creating the COBOL Copybook Document

You can add a COBOL copybook document to a project two ways. One is by creating a new empty file and adding the content; the second is by importing an existing copybook document into the project.

Follow the appropriate procedure below to add a COBOL copybook document to your project:

- "To Create a New Document" on page 7
- "To Import an Existing Document" on page 8

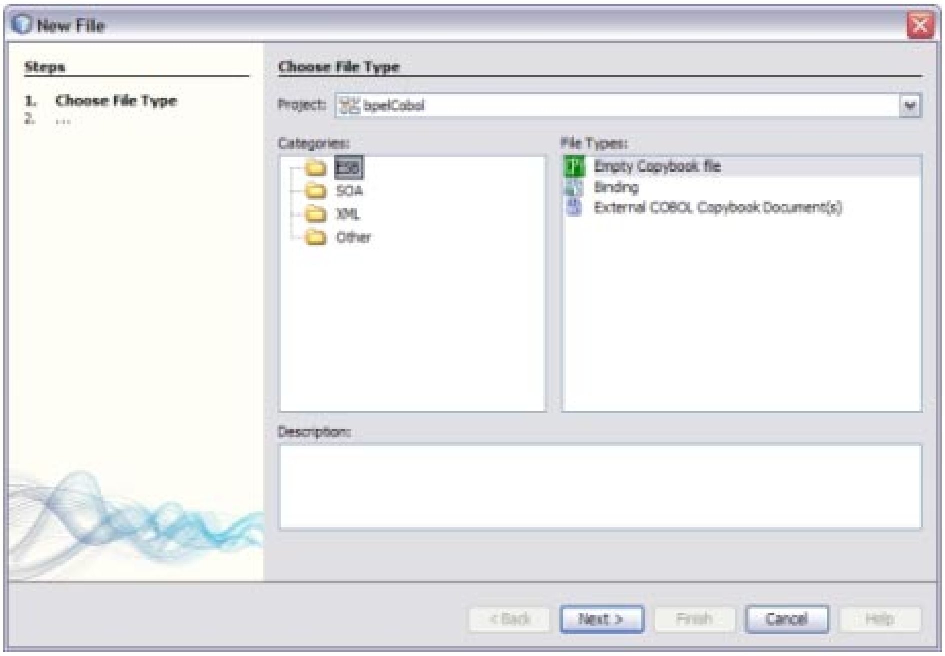
To Create a New Document

1 **In your NetBeans project, right-click the folder where you want to add the document (typically the Process Files folder).**

2 **Point to New and then select Other.**

The New File wizard appears.

3 **Under Categories, select ESB; under File Types, select Empty Copybook file.**



New File

Steps

1. Choose File Type
2. ...

Choose File Type

Project: bpelCobol

Categories: ESB

File Types: Empty Copybook file

Description:

4 **Click Next.**

The Name and Location window appears.

5 **Enter a name for the new copybook file and update the folder if needed.**

6 **Click Finish.**

The empty copybook file is added to the project tree and is opened in a NetBeans editor.

7 **Fill in the contents of the copybook file.**

Tip – You can use the copy and paste features of the editor to copy content from another file.

8 **Save and close the file.**

▼ To Import an Existing Document

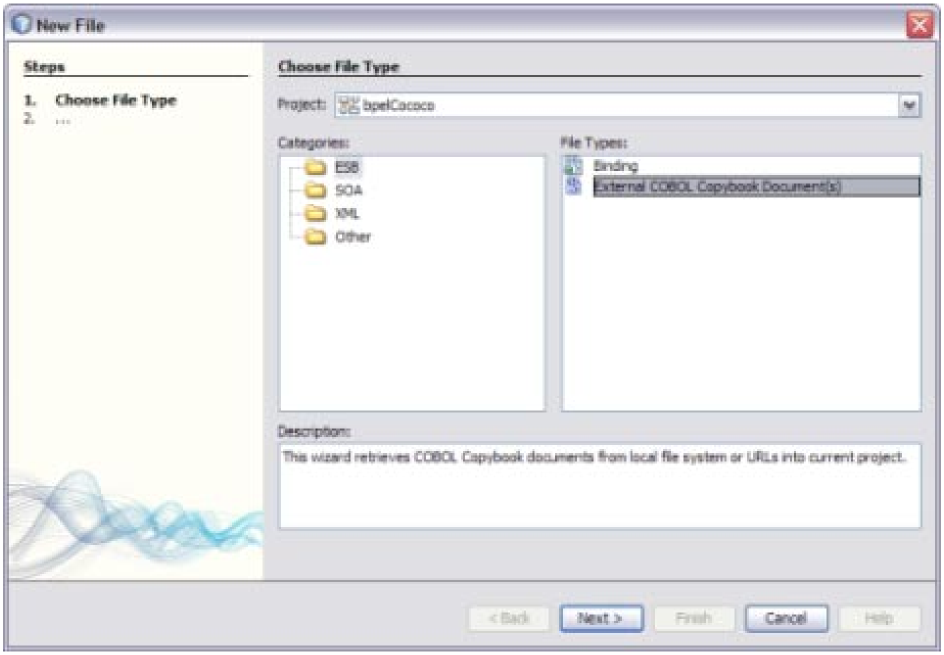
The COBOL Copybook Encoder supports the following file extensions: `CPY, COBOL, COB, CBL, CCC, and CCP`. If you have an external copybook document file with an extension other than those listed, rename the file by changing the extension to a supported extension.

1 **In your NetBeans project, right-click the folder where you want to add the document (typically the Process Files folder).**

2 **Point to New and then select Other.**

The New File wizard appears.

3 **Under Categories, select ESB; under File Types, select External COBOL Copybook Document(s).**



Project: bpelCococo

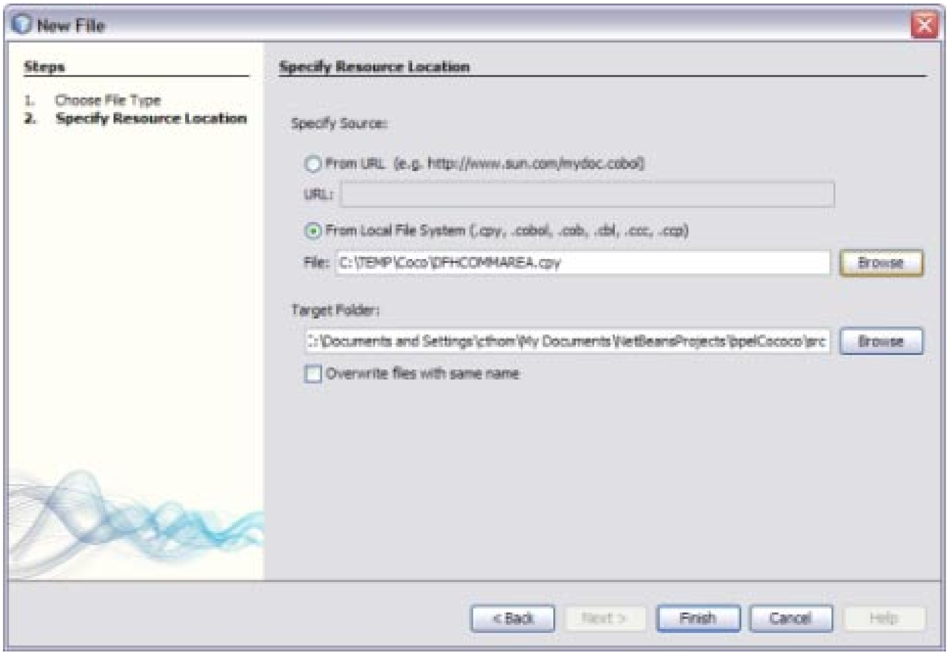
Categories: ESB

File Types: External COBOL Copybook Document(s)

Description: This wizard retrieves COBOL Copybook documents from local file system or URLs into current project.

4 Click Next.

The Specify Resource Location window appears.



New File

Steps

1. Choose File Type
2. **Specify Resource Location**

Specify Resource Location

Specify Source:

○ From URL (e.g. http://www.sun.com/mydoc.cobol)

URL:

● From Local File System (.cpy, .cobol, .cob, .cbl, .ccc, .ccp)

File: C:\TEMP\Coco\DFHCOMMAREA.cpy

Target Folder:

:\Documents and Settings\cthom\My Documents\NetBeansProjects\bpelCococo\src

□ Overwrite files with same name

5 Under Specify Source, do one of the following:

- **To access the copybook file from a URL, select From URL and then enter the URL in the URL field.**
- **To access the copybook file from your local file system, select From Local File System and then either enter the fully qualified path and file name or click Browse to navigate to and select the file.**

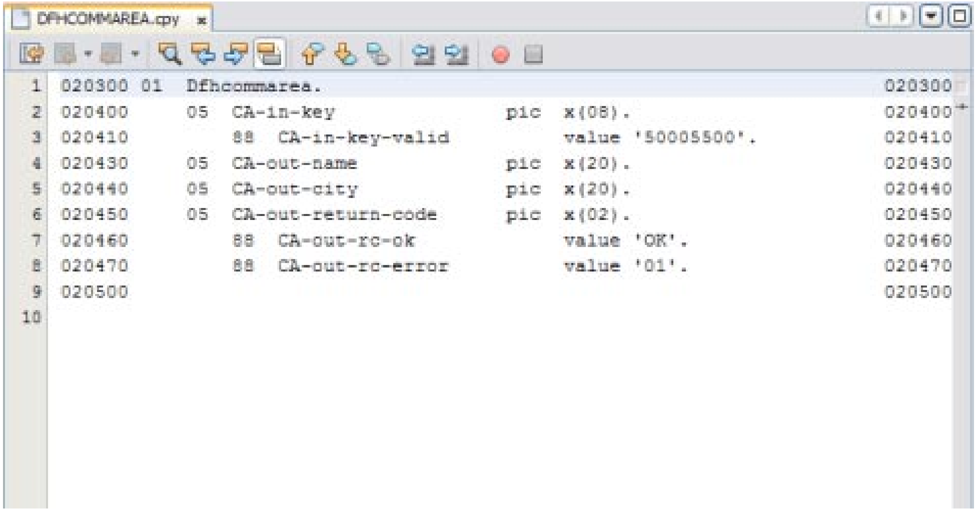
6 Accept or modify the default target folder.

The default location is the `src` directory in your NetBeans project.

7 To overwrite existing files, select the check box next to Overwrite Files With Same Name.

8 **Click Finish.**

The imported file appears in the project tree, and the content of the file appears in the editor.



```
020300 01  Dfhcommarea.                                              020300
020400     05  CA-in-key                pic  x(08).                  020400
020410         88  CA-in-key-valid          value '50005500'.        020410
020430     05  CA-out-name              pic  x(20).                  020430
020440     05  CA-out-city              pic  x(20).                  020440
020450     05  CA-out-return-code       pic  x(02).                  020450
020460         88  CA-out-rc-ok             value 'OK'.              020460
020470         88  CA-out-rc-error          value '01'.              020470
020500                                                               020500
```

Generating the XML Schema Definition (XSD)

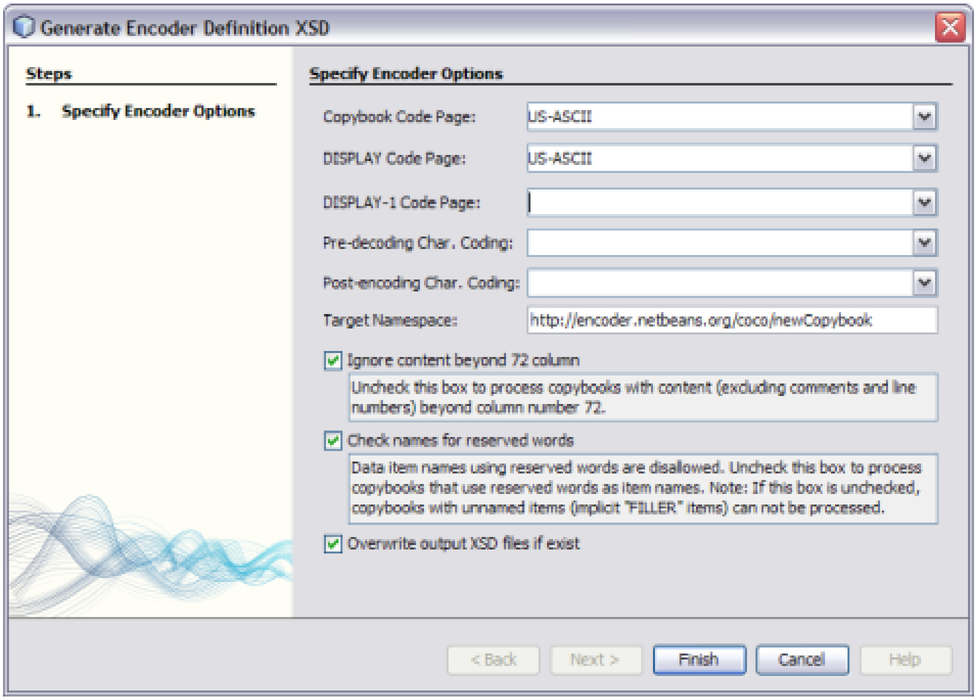
This step creates the annotated XSD file with information specific to the COBOL Copybook Encoder.

▼ To Generate the XSD File

1 In your NetBeans project, right-click the copybook file for which you want to generate the XSD file.

2 Select Generate Encoder Definition XSD.

The Generate Encoder Definition XSD Wizard appears.



Generate Encoder Definition XSD

Steps

1. Specify Encoder Options

Specify Encoder Options

Copybook Code Page: US-ASCII

DISPLAY Code Page: US-ASCII

DISPLAY-1 Code Page:

Pre-decoding Char. Coding:

Post-encoding Char. Coding:

Target Namespace: http://encoder.netbeans.org/coco/newCopybook

☑ Ignore content beyond 72 column

Uncheck this box to process copybooks with content (excluding comments and line numbers) beyond column number 72.

☑ Check names for reserved words

Data item names using reserved words are disallowed. Uncheck this box to process copybooks that use reserved words as item names. Note: If this box is unchecked, copybooks with unnamed items (implicit "FILLER" items) can not be processed.

☑ Overwrite output XSD files if exist

3 Enter the following information for the encoder definition:

- **Copybook Code Page**: The character encoding of the data in the copybook file.
- **DISPLAY Code Page**: The character encoding to use for data when the USAGE clause is DISPLAY.
- **DISPLAY-1 Code Page**: The character encoding to use for data when the USAGE clause is DISPLAY-1.
- **Pre-decoding Char. Coding**: The character encoding of the input data that needs to be converted to data that can be subsequently decoded by the COBOL Copybook Encoder.
- **Post-encoding Char. Coding**: The character encoding of the output data that is converted from the data used by the COBOL Copybook Encoder.
- **Target Namespace**: The target namespace for the XSD file.

4 **To process copybooks with content beyond column number 72 (excluding comments and line numbers), deselect the check box next to Ignore Content Beyond 72 Column; otherwise leave this check box selected.**

5 **To process copybooks that use reserved words as item names, deselect the check box next to Check names for Reserved Words; otherwise leave this check box selected.**

If this check box is not selected, copybooks with unnamed items (implicit FILLER items) cannot be processed.

6 **To make sure that no existing files of the same name are overwritten, deselect the check box next to Overwrite Output XSD Files if Exist; otherwise leave this check box selected.**

7 **Click Finish.**

The new XSD file is added to the project tree and the contents appears in the NetBeans editor.

Creating a WSDL Document to Use COBOL Copybook Encoding

When you create a concrete WSDL document for your project, you can specify COBOL copybook encoding for the data. Note that is only supported with a subset of binding components. The following procedure describes the process for the File Binding Component using a poll and write scenario. This illustrates using the encoder for both input and output files.

▼ To Create a WSDL Document for the File Binding Component

1 **In your NetBeans project, right-click the folder where you want to add the document (typically the Process Files folder).**

2 **Point to New and then select WSDL Document.**

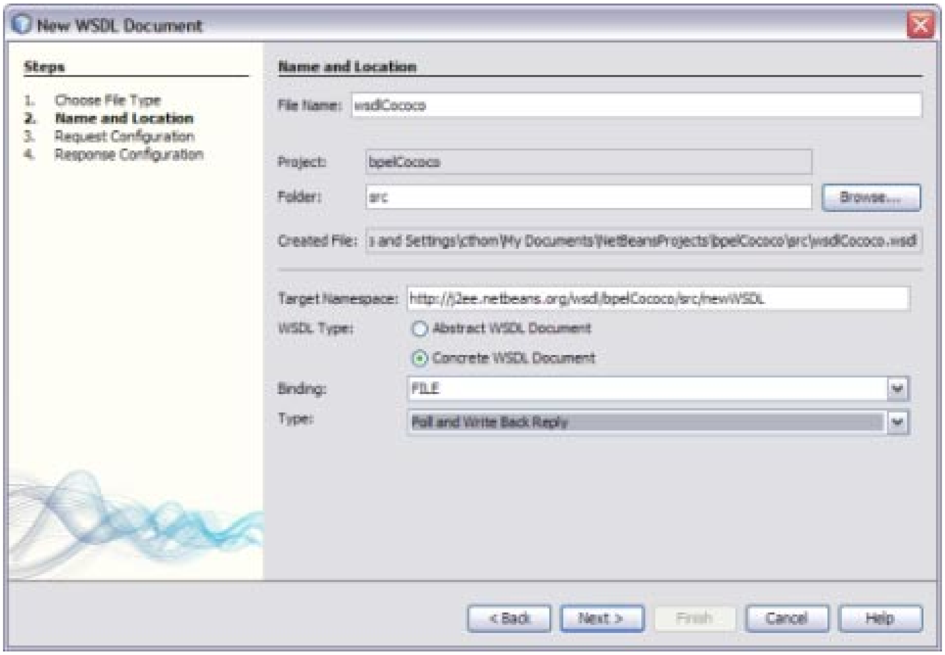
The New WSDL Document wizard appears.

3 **Enter a file name for the WSDL document, and accept the default folder location or specify a new one.**

4 **Under WSDL Type, select Concrete WSDL.**

The Binding and Type fields appear.

5 **Select the File binding, and then select the Poll and Write Back Reply type.**



File Name: wsdlCococo

Project: bpelCococo

Folder: src

Created File: s and Settings\cthom\My Documents\NetBeansProjects\bpelCococo\src\wsdlCococo.wsdl

Target Namespace: http://j2ee.netbeans.org/wsdl/bpelCococo/src/newWSDL

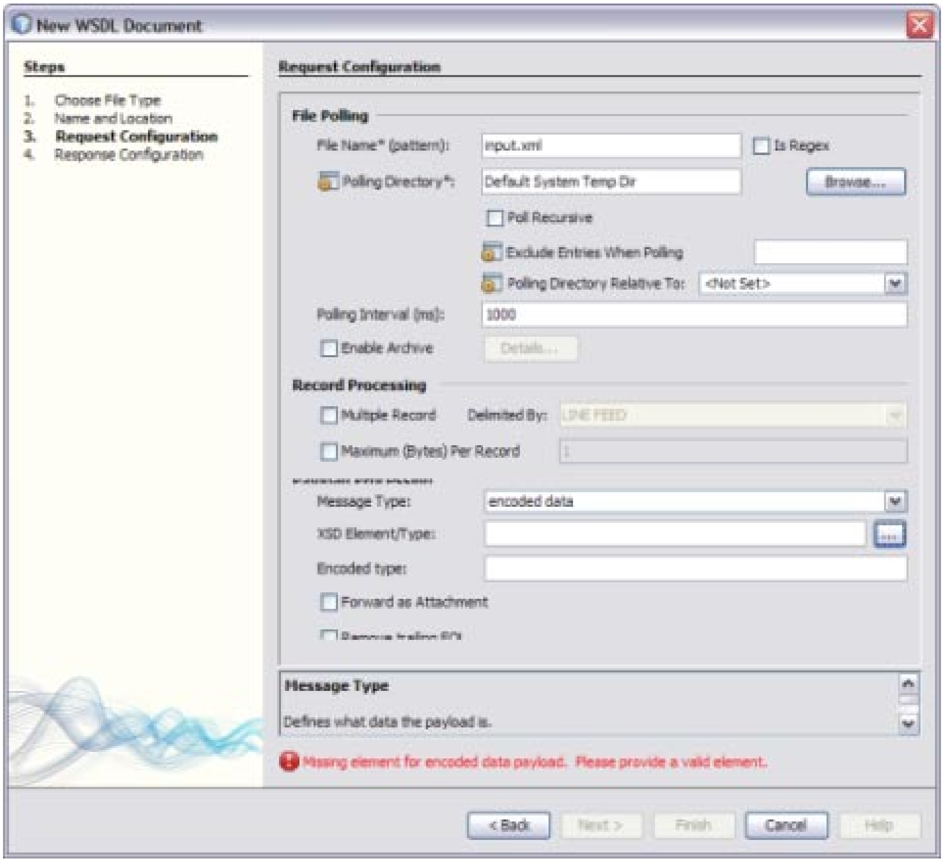
WSDL Type: ○ Abstract WSDL Document ● Concrete WSDL Document

Binding: FILE

Type: Poll and Write Back Reply

6 **Click Next.**

The Request Configuration window appears.



File Name* (pattern): input.xml □ Is Regex

Polling Directory*: Default System Temp Dir

□ Poll Recursive

Exclude Entries When Polling

Polling Directory Relative To: <Not Set>

Polling Interval (ms): 1000

□ Enable Archive

□ Multiple Record Delimited By: LINE FEED

□ Maximum (Bytes) Per Record 1

Message Type: encoded data

XSD Element/Type:

Encoded type:

□ Forward as Attachment

Message Type

Defines what data the payload is.

Missing element for encoded data payload. Please provide a valid element.

7 **Fill in the properties for the File Polling and Record Processing sections (see** *Oracle Java CAPS File Binding Component User's Guide*).

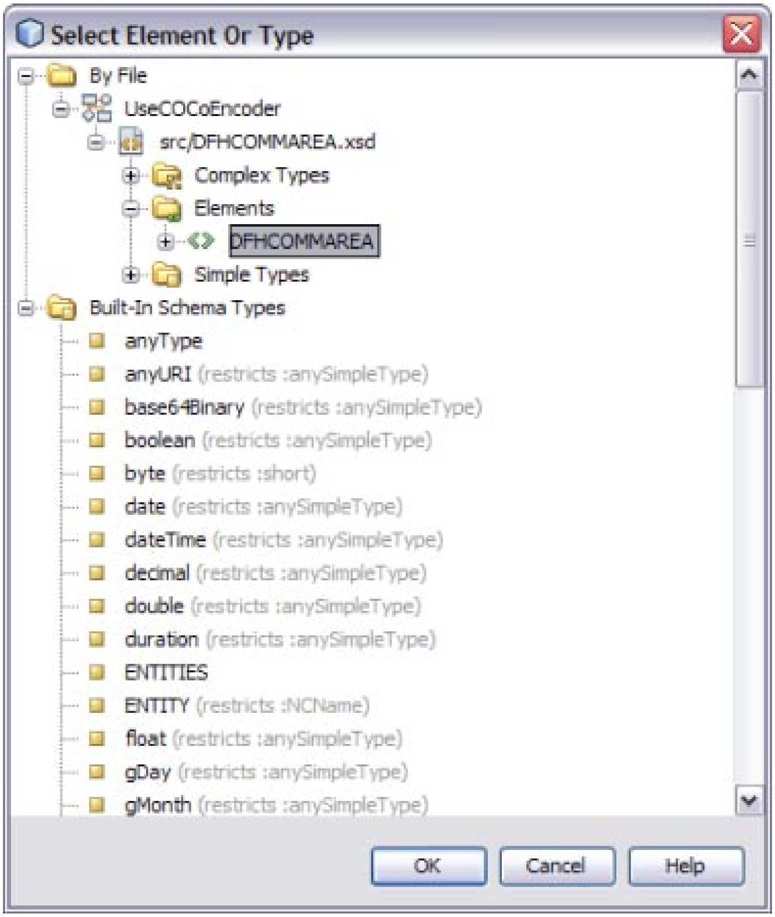
8 **In the Payload Processing section, do the following:**

a. **For the Message Type, select encoded data.**

The options beneath message type change.

b. **Click the ellipsis next to XSD Element/Type.**

The Select Element or Type dialog box appears.

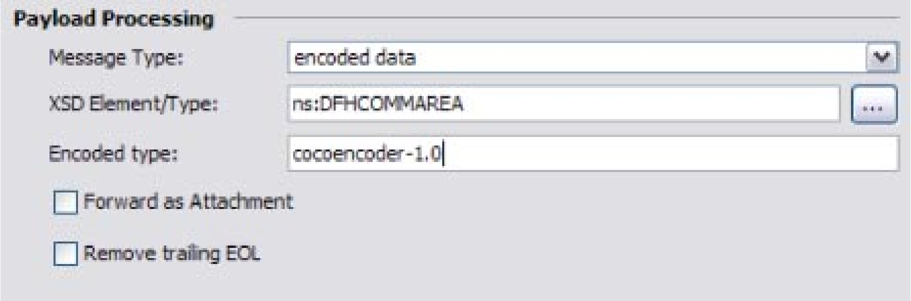

Select Element Or Type
By File
UseCOCoEncoder
src/DFHCOMMAREA.xsd
Complex Types
Elements
DFHCOMMAREA
Simple Types
Built-In Schema Types
anyType
anyURI (restricts :anySimpleType)
base64Binary (restricts :anySimpleType)
boolean (restricts :anySimpleType)
byte (restricts :short)
date (restricts :anySimpleType)
dateTime (restricts :anySimpleType)
decimal (restricts :anySimpleType)
double (restricts :anySimpleType)
duration (restricts :anySimpleType)
ENTITIES
ENTITY (restricts :NCName)
float (restricts :anySimpleType)
gDay (restricts :anySimpleType)
gMonth (restricts :anySimpleType)
OK
Cancel
Help

c. **Expand the By File node in the tree, and expand the copybook XSD file you created until you see the element to choose.**

d. **Click the Element and then click OK.**

e. **In the Encoded Type field on the Request Configuration page, enter `cocoencoder-1.0`.**

The Payload Processing section should be similar to the image below.



Payload Processing

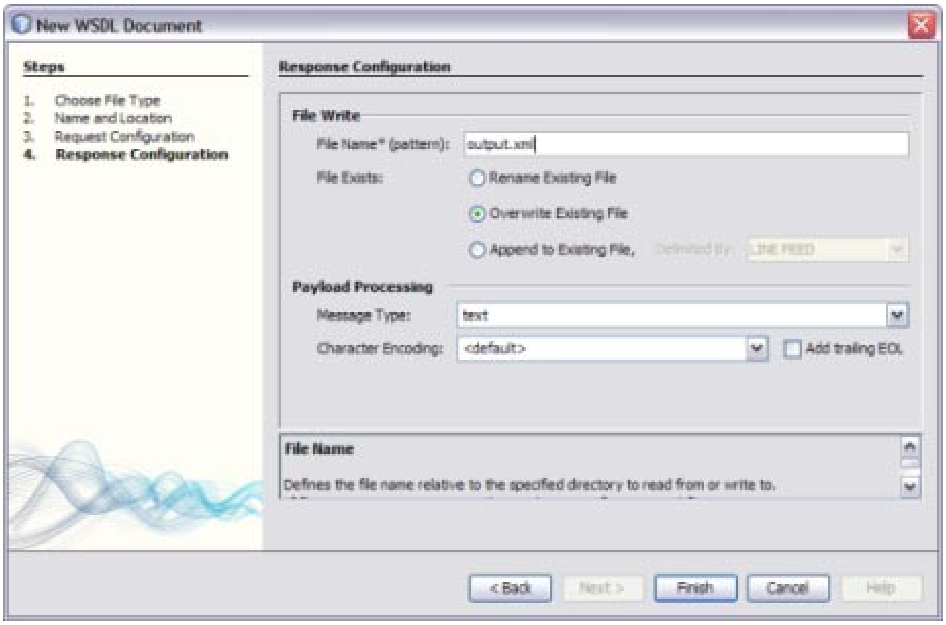
Message Type: encoded data

XSD Element/Type: ns:DFHCOMMAREA

Encoded type: cocoencoder-1.0

9 **Click Next.**

The Response Configuration window appears.



New WSDL Document

Steps

1. Choose File Type
2. Name and Location
3. Request Configuration
4. **Response Configuration**

Response Configuration

File Write

File Name* (pattern): output.xml

File Exists: ○ Rename Existing File ● Overwrite Existing File ○ Append to Existing File, Delimited By: LINE FEED

Payload Processing

Message Type: text

Character Encoding: <default> □ Add trailing EOL

File Name

Defines the file name relative to the specified directory to read from or write to.

10 **Fill in the File Write fields, and then do the following for the Payload Processing section:**

a. **In the Message Type field, select encoded data.**

The options under Message Type change.

b. **Click the Ellipses next to the XSD Element/Type field.**

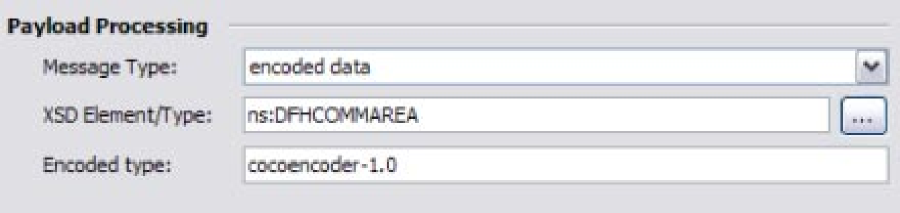
The Select Element or Type dialog box appears.

c. **Expand the By File node in the tree, and expand the copybook XSD file you created until you see the element to choose.**

d. **Click the Element and then click OK.**

e. **In the Encoded Type field on the Request Configuration page, enter `cocoencoder-1.0`.**

   The Payload Processing section should be similar to the image below.



Payload Processing

Message Type: encoded data

XSD Element/Type: ns:DFHCOMMAREA

Encoded type: cocoencoder-1.0

11 **Click Finish.**

The WSDL document appears in the project tree and its contents appears in the NetBeans WSDL Editor.

Applying COBOL Copybook Encoding to an Existing XSD

If you have an existing XSD that was not generated from a COBOL copybook document, you can apply copybook encoding to it.

▼ To Apply COBOL Copybook Encoding to an XSD

1 **Right-click the XSD to which COBOL copybook encoding will be applied.**

2 **Point to Encoding, and then select Apply COBOL Copybook Encoding.**

3 **On the dialog box that appears, deselect Apply to All Referenced Schemas or leave it selected, and then click Process.**

Once you apply the Encoder, a special encoding node is automatically added to XSD elements as a child node of an annotation node.

4 **After applying the Encoder, the context menu changes as shown below to allow you to do either of the following:**

- **To reset the encoding parameters to their default values (if you modified any), click Reapply COBOL Copybook Encoding. The node structure you have created will be preserved.**

- **To remove the COBOL copybook encoding, click Remove Encoding.**

5. **Once you apply the Encoder, you can define the encoding rules by adding elements specific to the COBOL Copybook Encoder to the annotation elements.**

Validating and Testing the COBOL Copybook Encoder

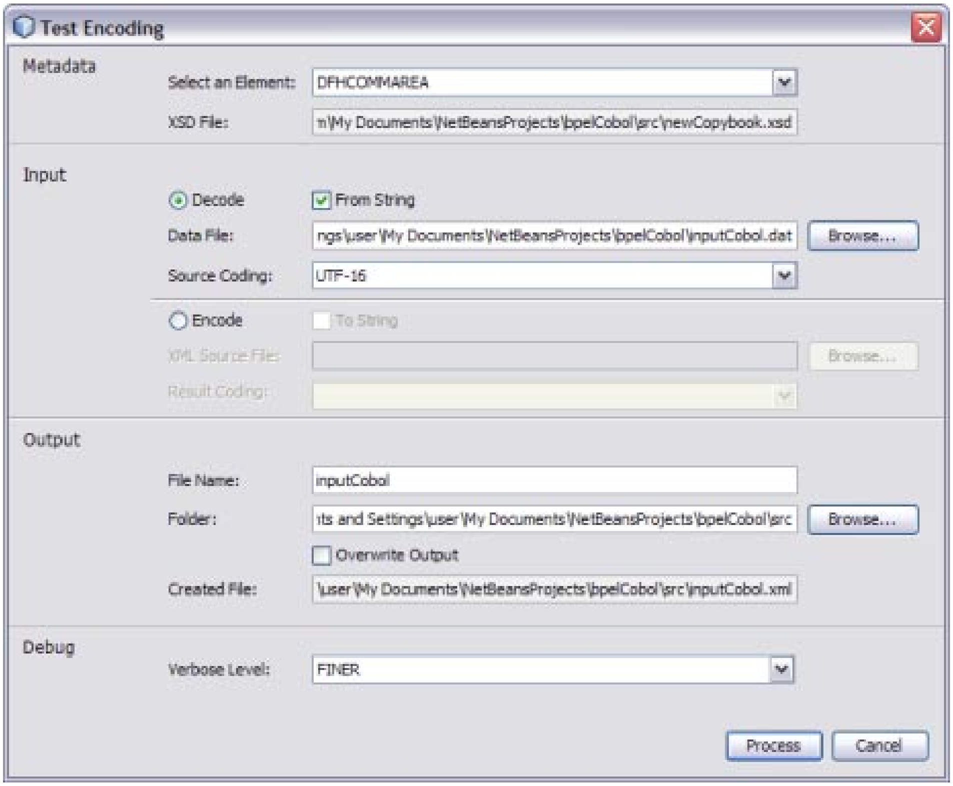
You can validate the encoding rules in the copybook's XSD file by opening the file in the XSD Editor and clicking the Validate File icon in the XSD Editor. The file is validated for XML grammar and semantics, and if encoding rules are present, they are validated as well.

You can test the Encoder's runtime behavior using the Encoder Tester. After the Decode test is complete, the result is placed in an XML file inside the current project.

▼ Testing the COBOL Copybook Encoder

1 **In the NetBeans Projects window, right-click the XSD file to test, point to Encoding and click Test.**

The Test Encoding dialog box appears.



Test Encoding

Metadata

Select an Element: DFHCOMMAREA

XSD File: n\My Documents\NetBeansProjects\bpelCobol\src\newCopybook.xsd

Input

● Decode ☑ From String

Data File: ngs\user\My Documents\NetBeansProjects\bpelCobol\inputCobol.dat

Source Coding: UTF-16

○ Encode □ To String

XML Source File:

Result Coding:

Output

File Name: inputCobol

Folder: nts and Settings\user\My Documents\NetBeansProjects\bpelCobol\src

□ Overwrite Output

Created File: \user\My Documents\NetBeansProjects\bpelCobol\src\inputCobol.xml

Debug

Verbose Level: FINER

2 **Fill in the fields for the type of test you want to perform.**

For more information, see the table at the end of this procedure.

3 **Click Process.**

When a Decode test is complete, the parsed XML file is generated and opened in the NetBeans editor. When an Encode test is complete, the output COBOL copybook native data is generated. Information on the testing process appears below the editor in the Output panel.

Tip – The generated XML file for the Decode test is not formatted. To format the file, click Source in the NetBeans toolbar, and then select Format from the drop-down menu.

Section	Field	Description
Metadata	Select an Element	The top-level element whose structure you want to test.
	XSD File	The XSD file you have selected for testing.
Input	Decode/Encode	Option buttons to select the direction of data flow for the test. These options specify whether encoding or decoding behavior is being tested.
	From/To String	An indicator that the input or output data is in string format. If not checked, byte format is assumed.
	Data File	The data file to use in a Decode test.
	XML Source File	The source file to use in an Encode test.
	Source/Result Coding	The encoding of the serialized data.
Output	File Name	The file name to use for the test result.
	Folder	The folder in which you want the output file to be placed.
	Overwrite Output	An indicator of whether or not to overwrite any existing output file having the same name.
	Created File	The path and file name of the output file.
Debug	Verbose Level	Specifies the level of detail contained in the log file. The options are: ■ `None` ■ `Info` ■ `Fine` ■ `Finer` ■ `Finest`