BEA WebLogic Java Adapter for Mainframe

Reference Guide

BEA WebLogic Java Adapter for Mainframe
Reference Guide 4.2
Document Edition 4.2
July 2001

Copyright

Restricted Rights Legend

Trademarks or Service Marks

BEA WebLogic Java Adapter for Mainframe Reference Guide

Document Edition	Part Number	Date	Software Version
4.2	N/A	July 2001	BEA WebLogic Java Adapter for Mainframe 4.2

Contents

About This Document

The BEA WebLogic Java Adapter for Mainframe product (hereafter referred to as JAM) is a gateway connectivity application that enables client/server interactions between Java applications and OS/390 Customer Information Control System/Enterprise System Architecture (CICS/ESA) or Information Management System (IMS) programs.

This document provides the following reference information to supplement the JAM documentation:

- "JAM Programming Reference" describes rules used by the eGen COBOL Code Generator.
- "eGen COBOL Code Generator Reference" contains reference pages for the BEA WebLogic Java Adapter for Mainframe eGen COBOL Code Generator.
- "Understanding How JAM Uses XML" describes XML and explains how JAM uses XML.
- "Security" describes supported security options for JAM.
- "Extracting Java Docs" describes how to extract the HTML pages that document the JAM Java classes
- "CRM Error Messages" describes the error, informational, and warning messages that can be encountered while using the CRM.
- "JAM Error Messages" describes the error, informational, and warning messages that can be encountered while using the JAM software
- "Index"

What You Need to Know

This document is intended for system administrators, application programmers, and business analysts who will use the BEA WebLogic Java Adapter for Mainframe application.

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation or go directly to the "e-docs" Product Documentation page at **http://edocs.bea.com/**.

How to Print the Document

A PDF version of this document is available on the JAM documentation Home page on the e-docs Web site (and also on the installation CD). You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format. To access the PDFs, open the JAM documentation Home page, click the PDF files button, and select the document you want to print.

If you do not have the Adobe Acrobat Reader, you can get it for free from the Adobe Web site at **http://www.adobe.com/**.

Related Information

The following BEA publications are available for JAM 4.2:

- *BEA WebLogic Java Adapter for Mainframe Release Notes*

- *BEA WebLogic Java Adapter for Mainframe Introduction*
- *BEA WebLogic Java Adapter for Mainframe Installation Guide*
- *BEA WebLogic Java Adapter for Mainframe Configuration and Administration Guide*
- *BEA WebLogic Java Adapter for Mainframe Programming Guide*
- *BEA WebLogic Java Adapter for Mainframe Scenarios Guide*
- *BEA WebLogic Java Adapter for Mainframe Workflow Processing Guide*
- *BEA WebLogic Java Adapter for Mainframe Reference Guide*

Contact Us

Your feedback on the BEA WebLogic Java Adapter for Mainframe documentation is important to us. Send us e-mail at **docsupport@bea.com** if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the JAM documentation.

In your e-mail message, please indicate that you are using the documentation for the BEA WebLogic Java Adapter for Mainframe 4.2 release.

If you have any questions about this version of JAM, or if you have problems installing and running JAM, contact BEA Customer Support through BEA WebSupport at **www.bea.com**. You can also contact Customer Support by using the contact information provided on the Customer Support Card that is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
blue text	Indicates a hypertext link in PDF or HTML
italics	Indicates emphasis or book titles or variables.
"string with quotes"	Indicates a string entry that requires quote marks.
UPPERCASE TEXT	Indicates generic file names, device names, environment variables, and logical operators. *Examples*: LPT1 SIGNON OR

monospace text	Indicates code samples, commands and their options, data structures and their members, data types, directories, and file names and their extensions. Monospace text also indicates text that you must enter from the keyboard.
	Examples:
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

monospace boldface text	Identifies significant words in code. *Example*: `void` **`xa_commit`** `( )`
{ }	Indicates a set of choices in a syntax line. The braces themselves should never be typed.

Convention	Item

`[ ]`	Indicates optional items in a syntax line. The brackets themselves should never be typed.
	Example:
	`buildclient [-v] [-o name ] [-f file-list]...`
	`[-l file-list]...`

|	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.

. . .	Indicates one of the following in a command line:
	■ That an argument can be repeated several times in a command line
	■ That the statement omits additional optional arguments
	■ That you can enter additional parameters, values, or other information
	The ellipsis itself should never be typed.
	Example:
	`buildclient [-v] [-o name ] [-f file-list]...`
	`[-l file-list]...`

. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

1 JAM Programming Reference

This section provides the rules that allow you to identify what form a generated Java class takes from a given COBOL copybook processed by the eGen COBOL Code Generator (eGen utility). An understanding of the rules facilitates a programmer's ability to correctly code any custom programs that make use of the generated classes.

The eGen utility maps a COBOL copybook into a Java class. The COBOL copybook contains a data record description. The eGen utility derives the generated Java class from the `com.bea.dmd.dataview.DataView` class (later referred to as `DataView`), which is provided on your BEA WebLogic Java Adapter for Mainframe (JAM) product CD-ROM in the `jam.jar` file.

This section discusses data mapping rules in the following topics:

You should find the COBOL terms in this section easy to understand; however, you may need to use a COBOL reference book or discuss the terms with a COBOL programmer. Also, you can process a copybook with the eGen utility and examine the generated Java code in order to understand the mapping.

Field Name Mapping Rules

When you process a COBOL copybook containing field names, they are mapped to Java names. The mapping is performed by the eGen utility according to the following rules:

1. All alphabetic characters are mapped to lower case, except in the following two cases.
2. All dashes are removed and the character following the dash is mapped to upper case.
3. When a prefix is added to the name (as when creating a field accessor function name), the first character of the base name is mapped to upper case.

Table 1-1 lists some mapping examples.

Table 1-1 Example Field Name Mapping from COBOL to Java and Accessor

COBOL Field Name	Java Base Name	Sample Accessor Name
`EMP-REC`	`empRec`	`setEmpRec`
`500-REC-CNT`	`500RecCnt`	`set500RecCnt`

Field Type Mappings

When you process a COBOL copybook, the data types of fields are mapped to Java data types. The mapping is performed by the eGen utility according to the following rules:

1. Groups map to `DataView` subclasses.
2. All alphanumeric fields are mapped to type `String`.
3. All edited numeric fields are mapped to type `String`.
4. All `SIGN SEPARATE`, `BLANK WHEN ZERO` or `JUSTIFIED RIGHT` fields are mapped to type `String`.
5. `SIGN IS LEADING` is not supported.
6. The types `COMP-1`, `COMP-2`, `COMP-5`, `COMP-X`, and `PROCEDURE-POINTER` fields are not supported (an error message is generated).
7. All `INDEX` fields are mapped to Java type `int`.
8. `POINTER` maps to Java type `int`.
9. All numeric fields with any digits to the right of the decimal point are mapped to type `BigDecimal`.
10. All `COMP-3` (packed) fields are mapped to type `BigDecimal`.
11. All other numeric fields are mapped as shown in Table 1-2.

Table 1-2 Numeric Field Mapping

Number of Digits	Java Type
<= 4	`short`
> 4 and <= 9	`int`
> 9 and <= 18	`long`
> 18	`BigDecimal`

Group Field Accessors

Each nested group in a COBOL copybook is mapped to a corresponding `DataView` subclass. The generated subclasses are nested exactly as the COBOL groups in the copybook. In addition, the eGen utility generates a private instance variable of this class type and a `get` accessor.

For example, the following copybook:

```
10 MY-RECORD.
       20  MY-GRP.
              30  ALNUM-FIELD             PIC X(20).
```

Produces code similar to the following:

```
public MyGrp2V getMyGrp();
public static class MyGrp2V extends DataView
{
              // Class definition
}
```

Elementary Field Accessors

Each elementary field is mapped to a private instance variable within the generated DataView subclass. Access to this variable is accomplished by two accessors that are generated (`set` and `get`).

These accessors have the following forms:

```
public void setFieldName(FieldType value);

public FieldType getFieldName();
```

Where:

`FieldType`
: is described in the "Field Type Mappings" section.

`FieldName`
: is described in the "Field Name Mapping Rules" section.

For example, the following copybook:

```
10 MY-RECORD.
       20  NUMERIC-FIELD                  PIC S9(5).
       20  ALNUM-FIELD                    PIC X(20).
```

Produces the accessors:

```
public void setNumericField(int value);
public int getNumericField();
public void setAlnumField(String value);
public String getAlnumField();
```

Array Field Accessors

Array fields are handled according to the field accessor rules described in "Group Field Accessors" and "Elementary Field Accessors", with the addition that each accessor takes an additional `int` argument that specifies which array entry is to be accessed, for example:

```
public void          set FieldName (int index, FieldType value);
public FieldType     get FieldName (int index);
```

Array fields specified with the `DEPENDING ON` clause are handled the same as fixed-size arrays with the following special rules:

1. The accessors may be used to `get` or `set` any instance up to the maximum array index.
2. The controlling (`DEPENDING ON`) variable is evaluated when the DataView is converted to or from an external format, such as a mainframe format. The eGen utility converts only the array elements with subscripts less than the controlling value.

Fields with REDEFINES Clauses

Fields that participate in a `REDEFINES` set are handled as a unit. A private `byte[]` variable is declared to hold the underlying mainframe data, as well as a private `DataView` variable. Each of the redefined fields has an accessor or accessors. These accessors take more CPU overhead than the normal accessors because they perform conversions to and from the underlying `byte[]` data.

For example the copybook:

```
10 MY-RECORD.
       20 INPUT-DATA.
              30 INPUT-A                                  PIC X(4).
              30 INPUT-B                                  PIC X(4).
       20 OUTPUT-DATA REDEFINES INPUT-DATA                PIC X(8).
```

Produces Java code similar to the following:

```
private byte[] m_redef23;
private DataView m_redef23DV;
public InputDataV   getInputData();
public String getOutputData();
public void   setOutputData(String value);
public static class InputDataV extends DataView
{
// Class definition.
}
```

COBOL Data Types

This section summarizes the COBOL data types supported by JAM software. Table 1-3 lists the COBOL data item definitions recognized by the eGen utility. Table 1-4 lists the syntactical features and data types recognized by the eGen utility. If a COBOL feature is unsupported and it is not listed as ignored in the table, an error message is generated.

Table 1-3 Major COBOL Features

COBOL Feature	Support
IDENTIFICATION DIVISION	Unsupported
ENVIRONMENT DIVISION	Unsupported
DATA DIVISION	Partially Supported
WORKING-STORAGE SECTION	Partially Supported
Data record definition	Supported
PROCEDURE DIVISION	Unsupported
COPY	Unsupported
COPY REPLACING	Unsupported
EJECT, SKIP1, SKIP2, SKIP3	Supported

Table 1-4 COBOL Data Types

COBOL Type	Java Type
COMP, COMP-4, BINARY *(integer)*	Short/Int/Long
COMP, COMP-4, BINARY *(fixed)*	BigDecimal
COMP-3, PACKED-DECIMAL	BigDecimal
COMP-5	Unsupported
COMP-X	Unsupported
`DISPLAY numeric (zoned)`	BigDecimal
`BLANK WHEN ZERO (zoned)`	String
SIGN IS LEADING *(zoned)*	Unsupported
`SIGN IS LEADING SEPARATE (zoned)`	String
`SIGN IS TRAILING (zoned)`	String

Table 1-4 COBOL Data Types

COBOL Type	Java Type
`SIGN IS TRAILING SEPARATE (zoned)`	String
edited numeric	String
`COMP-1, COMP-2 (float)`	Unsupported
edited float numeric	String
`DISPLAY (alphanumeric)`	String
edited alphanumeric	String
INDEX	Int
POINTER	Int
PROCEDURE-POINTER	Unsupported
JUSTIFIED RIGHT	Unsupported (ignored)
SYNCHRONIZED	Unsupported (ignored)
REDEFINES	Supported
66 RENAMES	Unsupported
66 RENAMES THRU	Unsupported
77 level	Supported
88 level *(condition)*	Unsupported (ignored)
group record	Inner Class
`OCCURS (fixed array)`	Array
`OCCURS DEPENDING (variable-length array)`	Array
OCCURS INDEXED BY	Unsupported (ignored)
OCCURS KEY IS	Unsupported (ignored)

Other Access Methods for Generated DataView Classes

JAM allows you to access DataView classes through several methods as described in the following sections:

- Mainframe Access to DataView Classes
- XML Access to DataView Classes
- Hashtable Access to DataView Classes

Mainframe Access to DataView Classes

This section describes how mainframe format data may be moved into and out of DataView classes. The eGen COBOL tool writes this code for you, so this information is provided as reference.

Mainframe format data may be extracted from a DataView class through the use of the `MainframeWriter` class. Listing 1-1 shows a sample of code that may be used to perform the extraction.

Listing 1-1 Sample Code for Extracting Mainframe Format Data from a DataView Class

```
import com.bea.base.io.MainframeWriter;
import com.bea.dmd.dataview.DataView;

   ...

   /**
    * Get mainframe format data from a DataView into a byte[].
    */
   byte[] getMainframeData(DataView dv)
   {
        try
        {
```

```
            MainframeWriter mw = new MainframeWriter();
            // To override the DataView's codepage, change the
            // above constructor call to something like:
            // ...new MainframeWriter("cp1234");

            return dv.toByteArray(mw);
        }
        catch (java.io.IOException e)
        {
            // Some conversion failure occurred…
        }
    }
```

If you wish to override the codepage provided when the DataView was generated, you may provide another codepage as a `String` argument to the `MainframeWriter` constructor, as shown in the comment in Listing 1-1.

Loading mainframe data into a DataView is a similar process, in this case requiring the use of the `MainframeReader` class. Listing 1-2 shows a sample of code that may be used to perform the load.

Listing 1-2 Sample Code for Loading Mainframe Data into a DataView Class

```
import com.bea.base.io.MainframeReader;
import com.bea.dmd.dataview.DataView;

    ...

    /**
     * Put a byte[] containing mainframe format data into a DataView.
      */
    MyDataView     putMainframeData(byte[] buffer)
    {
         MainframeReader mr = new MainframeReader(buffer);
         // To override the DataView's codepage, change the above
         // constructor call to something like:
         // …new MainframeReader("cp1234", buffer);
        .
        .
        .
    MyDataView dv;
        .
        .
```

```
    .
try
     {
      // Construct a new DataView with the mainframe data.
          dv = new MyDataView(mr);

      // Or, to load a pre-existing DataView with mainframe data.
      // dv.mainframeLoad(mr);
     }
     catch (java.io.IOException e)
     {
          // Some conversion failure occurred.
     }

     return dv;
}
```

XML Access to DataView Classes

Facilities are provided to move XML data into and out of DataView classes. These operations are performed through the use of the `XmlLoader` and `XmlUnloader` classes.

- `XmlLoader` is used to load XML data into a DataView.
- `XmlUnloader` is used to unload data from a DataView into XML.
- If the eGen COBOL script used to produce the DataView specifies the "support xml" option, then both a DTD and an XML/Schema that describe the XML format for this DataView are produced.

Listing 1-3 shows an example of the code used to load XML data into a DataView.

Listing 1-3 Sample Code for Loading XML Data into a DataView

```
import com.bea.dmd.dataview.DataView;
import com.bea.dmd.dataview.XmlLoader;

   ...

   void loadXmlData(String xml, DataView dv)
```

```
{
      XmlLoader xl = new XmlLoader();
      try
      {
         // Load the xml.  Note that the xml argument may be either
            // a String or a org.w3c.dom.Element object.
            xl.load(xml, dv);
      }
      catch (Exception e)
      {
            // Some conversion error occurred.
      }
}
```

Listing 1-4 shows an example of the code used to unload a DataView into XML.

Listing 1-4 Sample Code for Unloading a DataView into XML

```
import com.bea.dmd.dataview.DataView;
import com.bea.dmd.dataview.XmlUnloader;

   ...

   String unloadXmlData(DataView dv)
   {
         XmlUnloader xu = new XmlUnloader();

         try
         {
               String xml = xu.unload(dv);
               return xml;
         }
         catch (Exception e)
         {
               // Some conversion error occurred.
         }
   }
```

Hashtable Access to DataView Classes

JAM also provides facilities to load and unload DataView objects using Hashtable objects. Hashtable objects are most often used to move data from one DataView to another similar DataView.

When DataView fields are moved into Hashtables, each field is given a key that is a string reflecting the location of the field within the original copybook data structure. Listing 1-5 shows a sample of a COBOL Copybook.

Listing 1-5 Sample emprec.cpy COBOL Copybook

```
1       *-------------------------------------------------------
2       * emprec.cpy
3       *       An employee record.
4       *-------------------------------------------------------
5
6       02      emp-record.
7
8               04      emp-ssn                 pic 9(9)  comp-3.
9
10              04      emp-name.
11                      06      emp-name-last   pic x(15).
12                      06      emp-name-first  pic x(15).
13                      06      emp-name-mi     pic x.
14
15              04      emp-addr.
16                      06      emp-addr-street pic x(30).
17                      06      emp-addr-st     pic x(2).
18                      06      emp-addr-zip    pic x(9).
19
20      * End
```

The fields for the COBOL Copybook in Listing 1-5 are stored into a Hashtable as shown in the following table.

Key String	Content Type
empRecord.empSsn	BigDecimal

Key String	Content Type
empRecord.empName.empNameLast	String
empRecord.empName.empNameFirst	String
empRecord.empName.empNameMi	String
empRecord.empAddr.empAddrStreet	String
empRecord.empAddr.empAddrSt	String
empRecord.empAddr.empAddrZip	String

Code for Unloading and Loading Hashtables

Following is an example of the code used to **unload** a DataView into a Hashtable.

```
Hashtable ht = new HashtableUnloader().unload(dv);
```

Following is an example of the code used to **load** a Hashtable into an existing DataView.

```
new HashtableLoader().load(dv);
```

Rules for Unloading and Loading Hashtables

The basic rules of Hashtable **unloading** are:

- All data elements in the DataView are placed into the Hashtable.
- Each data item is stored as an object of its Java type. Elements of int/short/long type are converted to Integer/Short/Long.
- Arrays are mentioned at the appropriate level in the key as an index enclosed in "[", "]" pairs. For instance, if empAddr was an array, then one key into the Hashtable might be "empRecord.empAddr[2].empAddrStreet".

The basic rules of Hashtable **loading** are:

- All data elements in the DataView attempt to acquire a value from the Hashtable. If no matching key exists, the element retains its original value.

- Hashtable members of the wrong type result in a ClassCastException being thrown.

Name Translator Interface Facility

A name translator interface facility is available to provide Hashtable name mappings. Both `HashtableLoader` and `HashtableUnloader` provide a constructor that accepts an argument of type "com.bea.dmd.dataview.NameTranslator". Listing 1-6 shows how this interface is defined.

Listing 1-6 Name Translator Interface

```
//=============================================================
// NameTranslator.java
//     Name Translator interface.
//
// Copyright ©2000, BEA Systems, Inc., all rights reserved.

//-------------------------------------------------------------

   package com.bea.dmd.dataview;


/**************************************************************
* Name Translator interface.
* An interface for a 'functor' object that translates field names.
*
* @version $Revision: 1.1 $
* @author Copyright ©2000, BEA Systems, Inc., all rights reserved.
*/

   public interface NameTranslator
   {
       public String   translate(String input);
   }

// End NameTranslator.java
```

You can write classes that implement this interface for your application. These implementations are used to translate the key strings before the Hashtable is accessed.

Following are some useful implementations that are included in the JAM library:

Class Constructor	Purpose
NameFlattener()	Reduces the key to the portion following the final period character.
PrefixChanger(String old, String add)	Removes an old prefix & adds a new one.
PrefixChanger(String old)	Removes a prefix.

The `HashtableLoader`, `HashtableUnloader`, and the various name translator classes are included in the "com.bea.dmd.dataview" package.

Known Limitations

Following are some of the known limitations of this version of the JAM product.

- Continuation lines are not recognized in COBOL copybooks. This is only a problem for long character literals occurring within `VALUES` clauses. Comment out the relevant clause to fix the problem.
- COBOL copybooks with array (table) data items having an `OCCURS DEPENDING ON` clause must be structured so that the depending-on counter data item is not contained within the same group data item as the one containing the array.
- `USAGE` clauses on group data items in COBOL copybooks are not properly propagated to their subordinated member data items.

2 eGen COBOL Code Generator Reference

This section contains reference pages for the BEA WebLogic Java Adapter for Mainframe eGen COBOL Code Generator (eGen utility). This information includes the rules for writing the script file that controls the code generator.

eGen COBOL

The eGen utility maps a COBOL copybook into a Java class.

Synopsis

Invoke the utility with the following command:

```
java com.bea.jam.egen.EgenCobol scriptfile
```

where:

`java`
: is the name of the Java virtual machine executable in the Java Development Kit (JDK).

`com.bea.jam.egen.EgenCobol`
: is the full class name of the eGen utility.

scriptfile
: is the script file that controls the eGen utility. You must write this script file on an application-by-application basis. (See Listing 2-1 for an example).

If the JAM installation bin directory has been added to your path, then the eGen utility may also be invoked with the command:

```
egencobol scriptfile
```

Listing 2-1 Example of `scriptfile.egen`

```
### example script
#

view demo.CustomDataView from emprec.cpy

service demoService accepts CustomDataView returns CustomDataView

page demoPage "Demo Page"
{
    view demo.CustomDataView

    buttons
    {
              "Try It" service(demoService) shows demoPage
    }
}

servlet demo.DemoServlet shows demoPage
```

Script Syntax Reserved Words

The reserved words shown in Table 2-1 must be used as specified in the "Grammar" section.

Note: A reserved word can be used as an identifier if it is enclosed in either single or double quotation marks (refer to "General Rules").

Table 2-1 Reserved Words

accepts	buttons	class	client	codepage	ejb
from	generate	group	is	method	page
reset	returns	server	service	servlet	shows
support	view	xml			

General Rules

- The '#' character and all following characters on the same line are a comment. Use the '#' character to specify commented text.
- The character sequence "//" and all following characters on the same line are a comment. Use the "//" characters to specify commented text.
- The character sequence "/*" and all following characters until the occurrence of the sequence "*/" are a comment. Use the "/*" characters to specify commented text that extends beyond one line.
- White space (including newlines) is not significant, except when it is used to seperate tokens. White space includes newlines, carriage returns, tabs, spaces, etc.
- Any sequence of letters, digits, underscores, or periods is a word.
- Any word that does not match a reserved word is an identifier.
- Any sequence of characters is treated as an identifier if it is enclosed in either single or double quotes. This allows the use of reserved words and sequences that contain spaces.

Grammar

The eGen COBOL script grammar uses a modified Backus-Naur Form (BNF) syntax, which is used in many industry-standard software reference guides. BNF syntax specifies a context-free grammar. Reserved words are shown in bold. Comments are in italics preceded by a dash (—).

```
script:
       definition | script definition

fulldefinition:
       generate definition | definition

definition:
       viewdef | servicedef | servletdef | ejbdef | classdef |
       pagedef

viewdef:
       view viewname from copybook | viewdf viewmodifier

viewmodifier:
       codepage codepagename | support xml

servicedef:
       service servicename accepts fullViewname returns
       fullViewname

servletdef:
       servlet classname shows pagename

ejbdef:
       clientejb | serverejb

clientejb:
       client ejb classname ejbregistration { clientmethods }

serverejb:
       server ejb classname ejbregistration { servermethods }

classdef:
       client class classname { clientmethods }

pagedef:
        page pagename title { view viewname buttons { buttonlist } }

buttonlist:
        buttondef | buttonlist buttondef

buttondef:
       servicebutton | ejbbutton
```

```
clientmethods:
       clientmethoddef | clientmethods clientmethoddef

clientmethoddef:
        method methodname is servicename

servermethods:
        servermehtoddef | servermethods servermethodddef

servermethoddef:
        method methodname (fullviewname) returns fullviewname

servicebutton:
       buttonname service ( servicename ) shows pagename

ejbbutton:
       buttonname ejbmethod ( fullViewname ) returns fullViewname

viewname:
       classname

fullViewname:
       viewname | viewname [ codepagename ]

copybook:
       identifier
```

—An identifier that names a file containing a COBOL data definition.

```
servicename:
       identifier
```

—An identifier that matches a resource definition in your **`jcrmgw.cfg`** *file*

```
pagename:
       identifier
```

—An identifier that names a page definition.

```
codepagename:
       identifier
```

—The name of a codepage to be used for character translation to/from mainframe data formats. This must be a codepage supported by the JDK being used.

```
methodname:
       identifier
```

—The name to be given to a generated Java method.

```
classname:
       identifier
```

—An identifier that names a Java class, including any package name.

`ejbregistration:`
`identifier`
—The name that will be used to register the home interface for an EJB.

`title:`
`identifier`
—The title to be placed into the HTML generated for a page.

`buttonname:`
`identifier`
—A button name that will be used in the HTML generated for a page.

`ejbmethod:`
`identifier`
—An EJB classname and method specification that should look like this:
`package.ejbclass.method`
or
`ejbclass.method`

Results of Running the eGen COBOL Code Generator

- The specified COBOL copybook is parsed for each DataView definition (described in the "JAM Programming Reference" section of this guide) and a Java source file for the specified `DataView` class is generated in the current directory.

  If XML support was requested, then the following files are also produced:

  - *`viewname`*.dtd - DTD file
  - *`viewname`*.xsd - XML Schema file

- For each servlet definition, a Java source file is generated in the current directory for the specified class.
- For each client class definition, a Java source file is generated in the current directory for the specified class.
- For each EJB definition, three Java source files and a deployment descriptor text file are generated in the current directory. The names of the generated files are listed in Table 2-2.

Table 2-2 Generated Files for EJB Definitions

Name of File	Purpose
*classname*Home.java	EJB Home Interface
*classname*Bean.java	EJB Implementation class
classname.java	EJB Remote Interface
classname-jar.xml	EJB Deployment descriptor
wl-classname-jar.xml	WebLogic Deployment Info

3 Understanding How JAM Uses XML

BEA WebLogic Java Adapter for Mainframe uses the capabilities of XML to exchange data between different applications and operating systems. Understanding basic XML terms will help you to understand JAM's XML capabilities and how they are used.

This section discusses the following topics:

What is XML?

Extensible Markup Language, or XML, is a text format for exchanging data between different systems. It allows data to be described in a simple, standard, text-only format. Since the data is presented in a standard form, applications on disparate systems can interpret the data using simple text parsing tools, instead of having to interpret data in proprietary binary formats.

XML documents come in two varieties: data and metadata.

- XML Data Document

Data records can be converted into XML documents, which can then be transmitted to other applications. The XML data documents contain a single top-level entity (or tag) that represents the entire data record. Fields within the record are represented by other subordinate entities nested within the top-level entity. Each entity has a unique tag name, which corresponds to a field within the original data record. Each entity has content, which is the actual data value of the field. Entities may also have attributes, which are values attached to the entities that augment the normal content values. The XML data document file name ends with a .xml extension.

See Listing 3-2 for an example XML data document.

- XML Metadata

  Every XML document consists of a top-level entity, which in turn may be composed of subordinate entities. The structure of these entities, which included their tag names, the order in which they occur, the type and length of their content values, and their allowed attribute values, is described by a metadata definition. Metadata definitions can take the form of XML documents themselves. There are two standard formats for XML metadata documents: XML Document Type Definition (DTD) and XML Schema.

Document Type Definition

A Document Type Definition, or DTD, defines the legal building blocks of an XML document. It defines the document structure with a list of legal elements (tags). While XML provides an application independent way of sharing data, the DTD provides a common definition for interchanging data.

Your application can use a standard DTD to verify that data that you receive from the outside world is valid. You can also use a DTD to verify your own data.

The XML DTD file name ends with a .dtd extension.

See Listing 3-3 for an example XML DTD document.

XML Schema

A schema specifies the structure of an XML document and constraints on its content. While XML is the meta-language that provides the rules for defining tag languages, an XML Schema document is a formal specification of the grammar for a particular tag language. The schema defines the elements that can appear within the document and the attributes that can be associated with an element. It also defines the structure of the document: which elements are child elements of others, the sequence in which the child elements can appear, and the number of child elements. It defines whether an element is empty or can include text. The schema can also define default values for attributes.

XML Schema is more precise than DTD, providing more descriptive information about each XML element. It is likely that XML Schema will eventually replace XML DTD as the dominant standard metadata format.

A schema is useful for validating the document content to determine whether a document is a valid instance of the grammar expressed by that schema and for describing your grammar for use by others.

The XML Schema file name ends with a .xsd extension.

See Listing 3-4 for an example XML Schema document.

How JAM Uses XML

The JAM eGen COBOL Code Generator provides the ability to generate both XML Schema and XML DTD (Document Type Definition) documents for a given COBOL copybook record definition. The JAM runtime environment provides the capability of converting data records into XML data documents formatted according to their corresponding schema or DTD definitions.

The following listings show examples of the XML files generated by the eGen utility from the COBOL Copybook for an employee information record.

Listing 3-1 shows an example of an employee information record from a COBOL Copybook. The eGen utility generates an XML Schema and a DTD from the employee information record. Listing 3-2 shows the corresponding XML document that

conforms to the XML Schema and DTD generated from the employee record information, Listing 3-3 shows the corresponding DTD, and Listing 3-4 shows the corresponding XML Schema.

Listing 3-1 COBOL Copybook for Employee Information Record (emprec.cpy)

```
*-------------------------------------------------------
* emprec.cpy
* Employee record.
*
* @(#)$Id: emprec.cpy,v 1.2 1999/11/12 01:16:41 $
*-------------------------------------------------------
        02  emp-record.

        04  emp-ssn                    pic 9(9) comp-3.

        04  emp-name.
                06  emp-name-last      pic x(15).
                06  emp-name-first     pic x(15).
                06  emp-name-mi        pic x.

        04  emp-addr.
                06  emp-addr-street    pic x(30).
                06  emp-addr-st        pic x(2).
                06  emp-addr-zip       pic x(9).

* End
```

Listing 3-2 Example XML Document that Conforms to a DTD and XML Schema Generated from the eGen COBOL Code Generator (emprec.xml)

```
<emprec>
   <empRecord>
     <empSsn>660337645</empSsn>
     <empName>
       <empNameLast>Doe</empNameLast>
       <empNameFirst>John</empNameFirst>
       <empNameMi>P</empNameMi>
     </empName>
     <empAddr>
       <empAddrStreet>3235 Possum Park Ln.</empAddrStreet>
       <empAddrSt>TX</empAddrSt>
       <empAddrZip>758050000</empAddrZip>
```

```
     </empAddr>
   </empRecord>
</emprec>
```

Listing 3-3 DTD Generated from eGen COBOL Code Generator (emprec.dtd)

```
<!--
! DTD emprec 1.0
!
! Definition:   emprec
! Version:      1.0
! Source:       ../symbol/emprec.cpy
! Generated:    2000-09-27T19:18:25.084Z
! Created:      2000-09-27T19:18:24.937Z
! Modified:     1999-11-12T01:16:41.000Z
!-->

<!ELEMENT emprec
 ( empRecord )>

<!ATTLIST emprec
  date CDATA #DEFAULT "unknown">
  <!-- format="ccyy-mm-ddThh:mm:ss.mmmZ" -->

<!ATTLIST emprec
  version CDATA #DEFAULT "1.0">

<!-- empRecord -->
<!ELEMENT empRecord
 ( empSsn ,
   empName ,
   empAddr )>


<!-- empRecord.empSsn -->
<!ELEMENT empSsn
  (#PCDATA)>

<!-- empRecord.empName -->
<!ELEMENT empName
 ( empNameLast ,
   empNameFirst ,
   empNameMi )>

<!-- empRecord.empName.empNameLast -->
```

```
<!ELEMENT empNameLast
  (#PCDATA)>

<!-- empRecord.empName.empNameFirst -->
<!ELEMENT empNameFirst
  (#PCDATA)>

<!-- empRecord.empName.empNameMi -->
<!ELEMENT empNameMi
  (#PCDATA)>

<!-- empRecord.empAddr -->
<!ELEMENT empAddr
 ( empAddrStreet ,
   empAddrSt ,
   empAddrZip )>

<!-- empRecord.empAddr.empAddrStreet -->
<!ELEMENT empAddrStreet
  (#PCDATA)>

<!-- empRecord.empAddr.empAddrSt -->
<!ELEMENT empAddrSt
  (#PCDATA)>

<!-- empRecord.empAddr.empAddrZip -->
<!ELEMENT empAddrZip
  (#PCDATA)>

<!-- End -->
```

Listing 3-4 XML Schema Generated from eGen COBOL Code Generator (emprec.xsd)

```
<?xml version="1.0"?>
<schema
   xmlns:xsd="http://www.w3.org/1999/XMLSchema">

  <xsd:annotation>
    <xsd:documentation>
      Schema:       emprec
      Version:      1.0
      Source:       ../symbol/emprec.cpy
      Generated:    2000-09-27T19:19:42.857Z
      Created:      2000-09-27T19:19:43.708Z
```

```
    Modified:     1999-11-12T01:16:41.000Z
  </xsd:documentation>
</xsd:annotation>

<xsd:element name="emprec">
  <xsd:complexType>

    <xsd:attribute name="date"
       type="xsd:timeInstant"/>

    <xsd:attribute name="version"
       type="xsd:string"
       use="default"
       value="1.0"/>

    <xsd:element name="empRecord">
      <xsd:complexType>

        <xsd:element name="empSsn">
          <xsd:simpleType base="xsd:integer">
            <xsd:precision value="9"/>
            <xsd:minInclusive value="0">
          </xsd:simpleType>
          <!-- <%picture value="9(9)"/> -->
        </xsd:element>

        <xsd:element name="empName">
          <xsd:complexType>

            <xsd:element name="empNameLast"
               type="xsd:string"
               length="15"/>
               <!-- <%picture value="x(15)"/> -->

            <xsd:element name="empNameFirst"
               type="xsd:string"
               length="15"/>
               <!-- <%picture value="x(15)"/> -->

            <xsd:element name="empNameMi"
               type="xsd:string"
               length="1"/>
               <!-- <%picture value="x"/> -->

          </xsd:complexType>
        </xsd:element> <!--"empName"-->

        <xsd:element name="empAddr">
          <xsd:complexType>
```

```
                <xsd:element name="empAddrStreet"
                   type="xsd:string"
                   length="30"/>
                   <!-- <%picture value="x(30)"/> -->

                <xsd:element name="empAddrSt"
                   type="xsd:string"
                   length="2"/>
                   <!-- <%picture value="x(2)"/> -->

                <xsd:element name="empAddrZip"
                   type="xsd:string"
                   length="9"/>
                   <!-- <%picture value="x(9)"/> -->

             </xsd:complexType>
          </xsd:element> <!--"empAddr"-->

       </xsd:complexType>
    </xsd:element> <!--"empRecord"-->


  </xsd:complexType>
</xsd:element> <!--"emprec"-->

</schema>
```

4 Security

BEA WebLogic Java Adapter for Mainframe (JAM) supports the basic Application Program-to-Program Communication (APPC) style of sign-on security. You can configure a gateway to use one of three types of sign-on security for each link that is defined. The security options are defined in the `JC_LINKS` section of the `jcrmgw.cfg` file. Refer to the *BEA WebLogic Java Adapter for Mainframe Configuration and Administration Guide* for more information. The selected level of security determines which combination of user ID and password is used for transactions across the link.

Supported Security Options

JAM supports the following security options:

- `LOCAL`
  All security is handled by the local system and the link itself has no security requirement.
- `IDENTIFY`
  A user ID is passed to the mainframe. This user ID can originate with the client application or it can be a default user ID supplied at Java gateway startup by the `-u` option.
- `VERIFY`
  A user ID and password are passed to the mainframe. The user ID can originate with the client application or it can be a default user ID supplied at Java gateway startup by the `-u` option. The password must be supplied by the client application.

Note: For more information about the startup class, refer to the "Configuring the Gateway" section of the *BEA WebLogic Java Adapter for Mainframe Configuration and Administration Guide*.

In addition, an alternate mirror transaction is supported on each Distributed Program Link (DPL). The mirror transaction can be used to associate different Resource Access Control Facility (RACF) profiles with different services.

Refer to IBM RACF documentation for more specific information about establishing and administrating mainframe security.

Controlling User IDs and Passwords through Business Logic or Client Classes

User IDs and passwords used for mainframe requests can be controlled from business logic within client EJBs or from normal client classes. In order for this security credential control to work, you must have your gateway security level set to VERIFY.

Note: Following are the limitations of JAM security credential control:

- Individual control of security credentials from servlets that directly invoke services is not available.
- Encryption of the TCP/IP link between the WebLogic server and the CRM is not supported. Passwords are passed as clear text.
- Encryption of the SNA link between the CRM and the mainframe is not supported. Passwords are passed as clear text.
- Secure requests from the mainframe are not supported.

Controlling Security Credentials from Client EJB Code

Business logic within client EJB code can be used to control the security credentials used for mainframe requests. Client EJB implementations generated by eGen COBOL provide two routines, `setUserid()` and `setPassword()`, to adjust the user ID and password parameters.

The following listings are based upon the samples provided with JAM. They demonstrate the use of the `setUserid()` and `setPassword()` routines.

Listing 4-1 shows an eGen COBOL script used to generate a client EJB.

Listing 4-1 eGen COBOL Script to Generate a Client EJB

```
#-------------------------------------------------------------------------
# empclient.egen
# JAM script for a client EJB application.
#
# $Id: empclient.egen,v 1.1 2000/01/21 22:02:40  Exp $
#-------------------------------------------------------------------------

# Dataviews (typed data records)

view sample.EmployeeRecord from emprec.cpy

# Services

service sampleCreate
    accepts EmployeeRecord returns EmployeeRecord

service sampleRead
    accepts EmployeeRecord returns EmployeeRecord

service sampleUpdate
    accepts EmployeeRecord returns EmployeeRecord

service sampleDelete
    accepts EmployeeRecord returns EmployeeRecord

# Clients and servers

client ejb sample.SampleClient my.sampleBean
{
    method newEmployee
```

```
        is service sampleCreate

    method readEmployee
        is service sampleRead

    method updateEmployee
        is service sampleUpdate

    method deleteEmployee
        is service sampleDelete
}
```

When this script is passed to eGen COBOL, several files are generated, one of which is **`SampleClientBean.java`**, the EJB implementation. Listing 4-2 shows an example of this file.

Listing 4-2 Example of SampleClientBean.java File

```
// SampleClientBean.java
//
// EJB generated by EgenCobol on Dec 6, 2000.
//

package sample;

// Imports
//
import java.io.IOException;
import com.bea.jam.egen.EgenClientBean;
import com.bea.sna.jcrmgw.snaException;
import com.bea.base.io.MainframeWriter;
import com.bea.base.io.MainframeReader;

/**
 * EJB implementation.
 */
public class SampleClientBean extends EgenClientBean
{
    // deleteEmployee
    //
    public sample.EmployeeRecord deleteEmployee(sample.EmployeeRecord commarea)
        throws IOException, snaException
    {
        // Make the remote call.
```

```
    //
    byte[] inputBuffer = commarea.toByteArray(new MainframeWriter());
    byte[] rawResult = callService("sampleDelete", inputBuffer);
    sample.EmployeeRecord result =
         new sample.EmployeeRecord(new MainframeReader(rawResult));
    return result;
}

// updateEmployee
//
public sample.EmployeeRecord updateEmployee(sample.EmployeeRecord commarea)
    throws IOException, snaException
{
    // Make the remote call.
    //
    byte[] inputBuffer = commarea.toByteArray(new MainframeWriter());
    byte[] rawResult = callService("sampleUpdate", inputBuffer);
    sample.EmployeeRecord result =
         new sample.EmployeeRecord(new MainframeReader(rawResult));
    return result;
}

// readEmployee
//
public sample.EmployeeRecord readEmployee(sample.EmployeeRecord commarea)
    throws IOException, snaException
{
    // Make the remote call.
    //
    byte[] inputBuffer = commarea.toByteArray(new MainframeWriter());
    byte[] rawResult = callService("sampleRead", inputBuffer);
    sample.EmployeeRecord result =
         new sample.EmployeeRecord(new MainframeReader(rawResult));
    return result;
}

// newEmployee
//
public sample.EmployeeRecord newEmployee(sample.EmployeeRecord commarea)
    throws IOException, snaException
{
    // Make the remote call.
    //
    byte[] inputBuffer = commarea.toByteArray(new MainframeWriter());
    byte[] rawResult = callService("sampleCreate", inputBuffer);
    sample.EmployeeRecord result =
         new sample.EmployeeRecord(new MainframeReader(rawResult));
    return result;
}
```

```
}

// END SampleClientBean.java
```

Note that the four service routines all invoke the `callService` method to perform their work. Listing 4-3 illustrates a class that extends the generated EJB implementation to provide security credentials to the gateway during these operations.

Listing 4-3 Example of Class with Security Credentials

```
// ExtClientBean.java
//

package sample;

// Imports
//
import java.io.IOException;
import com.bea.sna.jcrmgw.snaException;

/**
 * EJB implementation.
 */
public class ExtClientBean extends SampleClientBean
{
    protected byte[] callService(String svc, byte[] input)
            throws snaException, IOException
    {
            setUserid("JAMUSER");
            setPassword("JAMPASS");

            return super.callService(svc, input);
    }
}

// END ExtClientBean.java
```

In order to deploy the extended EJB, the XML deployment descriptor must be edited to modify the `ejb-class` field. Listing 4-4 illustrates this file with the class name that must be changed marked in bold.

Listing 4-4 Extended EJB with Modified XML Deployment Descriptor

```
<?xml version="1.0"?>
<!DOCTYPE ejb-jar PUBLIC '-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans
1.1//EN' 'http://java.sun.com/j2ee/dtds/ejb-jar_1_1.dtd'>
<ejb-jar>
    <enterprise-beans>
      <session>
        <ejb-name>SampleClient</ejb-name>
        <home>sample.SampleClientHome</home>
        <remote>sample.SampleClient</remote>
        <ejb-class>sample.ExtClientBean</ejb-class>
        <session-type>Stateless</session-type>
        <transaction-type>Container</transaction-type>
      </session>
    </enterprise-beans>
    <assembly-descriptor>
      <container-transaction>
   <method>
     <ejb-name>SampleClient</ejb-name>
     <method-intf>Remote</method-intf>
     <method-name>*</method-name>
   </method>
   <trans-attribute>NotSupported</trans-attribute>
      </container-transaction>
    </assembly-descriptor>
</ejb-jar>
```

Controlling Security Credentials from Client Class Code

Client classes generated by eGen COBOL may be extended to specify security credentials used for requests. The parent class for generated client code provides `setUserid()` and `setPassword()` routines with the same signatures as those in the EJB model. These may be extended in the same manner. Refer to "Controlling Security Credentials from Client EJB Code" for examples of the EJB model code.

5 Extracting Java Docs

The BEA WebLogic Java Adapter for Mainframe (JAM) product comes with HTML pages that document the JAM Java classes. These also are referred to as "javadoc" files. They are located in the `jamdoc.jar` file, found in the JAM installation directory.

To view the contents of the javadoc HTML files from the .jar file without extracting it, issue the following command:

```
jar -tvf jamdoc.jar
```

Or,

To extract the javadoc HTML files from the .jar file, issue the following command:

```
jar -xvf jamdoc.jar
```

where:

`jar`
: is the Java archive command.

`-t`
: is the display Table of Contents parameter.

`-x`
: is the extract file(s) parameter.

`v`
: is the verbose parameter to list the files.

`f`
: is the option that designates the next parameter as the jar file name.

`jamdoc.jar`
: is the name of the JAM javadoc file.

This command extracts all of the files contained in the jar file into the current directory. The HTML documentation files are placed in a newly created subdirectory named `classdocs` in the current directory.

To view an HTML documentation file, open your web browser and specify the file name of the javadoc you want to view, taken from the `classdocs` directory. Any of the following files are good for getting started:

- `classdocs/AllNames.html`
- `classdocs/packages.html`
- `classdocs/tree.html`

6 CRM Error Messages

The following table contains a description of error, informational, and warning messages that can be encountered while using the Communications Resource Manager (CRM).

9001:ERROR	<taskname> timed out with failCode <failcode>	
	DESCRIPTION	A conversation has timed out in the CRM with the stack return code of <failcode>. A timer event set to watch a conversation has expired. <taskname> may appear as: OB-Conversation #*nn* (<linkref>) tx #*m* <tranname>, or IB-Conversation #nn (<linkref>) tx #*m* <tranname> where: *nn* is an internal APPC conversation number. *m* is the transaction context where -1 signifies non-transactional.
	ACTION	Examine stderr and the ULOG for additional information concerning the failure.
9002:ERROR	Server (<stackref>) Creation Failed	
	DESCRIPTION	CRM was unable to instantiate the stack object due to an error.
	ACTION	Check for additional messages in stderr. The shared library for the stack or the stack interface might not have been loaded due to an incorrect library path.

9003:ERROR	Server Failed (<stackref>), Code = <returncode>	
	DESCRIPTION	CRM received a bad return code from the stack start-up.
	ACTION	The <returncode> is the value returned by the SNA Stack software. Check the status of the stack, the configuration of the stack, and the gateway configuration. .
9004:ERROR	Configuration change on link <linkref> requires cold start	
	DESCRIPTION	Attempting to do a warm start after changing the domain configuration.
	ACTION	Change start type to COLD and restart.

9010:ERROR	<taskname> failed with failCode <failcode>	
	DESCRIPTION	A conversation has failed with the stack return code of <failcode>. <taskname> may appear as: OB-Conversation #*nn* (<linkref>) tx #*m* <tranname>, or IB-Conversation #nn (<linkref>) tx #*m* <tranname> where: *nn* is an internal APPC conversation number. *m* is the transaction context where -1 signifies non-transactional. Possible values for the <failcode> are: 1. Communications - unable to create the APPCserver object. 2. MemoryAllocation - internal error allocating memory. 3. InvalidObject - a CRM object could not be created or has been made invalid by some previous error. 4. InputOutput - error occurred during file I/O or an unexpected APPC return code was received. 5. Registration - internal task cannot be registered.
	ACTION	Examine stderr and the ULOG for additional information concerning the failure. For failcode Input/Output, verify that the user starting the CRM process has the proper file permissions for the BLOBLOG and RSTRTLOG. If no apparent error is found, contact BEA Customer Support.

9011:ERROR	Attempt to connect as second master refused!	
	DESCRIPTION	A second JAM gateway is attempting to connect to the CRM as a master gateway. Only one master gateway is allowed.
	ACTION	Ensure that multiple configurations do not use the same CRM address.
9012:ERROR	Attempt to connect as master in autonomous mode refused!	
	DESCRIPTION	An attempt to connect to the CRM as a master gateway was made when the CRM was running in autonomous mode.
	ACTION	Ensure that multiple configurations do not use the same CRM address.
9013:ERROR	Attempt to connect with incorrect group name (<groupname>) refused!	
	DESCRIPTION	The group name in the gateway configuration file does not match the group name specified in the CRM command line.
	ACTION	Correct the group name that is in error and restart.
9014:ERROR	INTERNAL ERROR: memory allocation failed [for new context/data buffer]	
	DESCRIPTION	Internal error allocating memory. No more memory.
	ACTION	Contact BEA Customer Support.
9015:ERROR	INTERNAL ERROR: server registration failed	
	DESCRIPTION	Internal error registering the APPC server. APPC libraries not found. The stack failed.
	ACTION	Contact BEA Customer Support.

9016:ERROR	Link refers to undefined APPC stack (<stackref>)!	
	DESCRIPTION	The stackref in the link configuration is incorrect.
	ACTION	Correct the stackref that is in error.
9017:ERROR	INTERNAL ERROR: link registration failed	
	DESCRIPTION	Internal error registering the link. The stack failed.
	ACTION	Contact BEA Customer Support.
9019:ERROR	Unknown Service Correlator = <correlator>, message dropped	
	DESCRIPTION	Internal error assigning service correlator values. Message context lost.
	ACTION	Contact BEA Customer Support.
9020:ERROR	Duplicate Service Correlator = <correlator>	
	DESCRIPTION	Internal error assigning service correlator values.
	ACTION	Contact BEA Customer Support.
9023:ERROR	Unknown Service Correlator = <correlator>, message dropped	
	DESCRIPTION	Internal error assigning service correlator values. Message context lost.
	ACTION	Contact BEA Customer Support.
9025:ERROR	Invalid Input Message Discarded	
	DESCRIPTION	Internal error, bad message sent between JAM gateway and CRM. Possibly incompatible JAM gateway and CRM.
	ACTION	Contact BEA Customer Support.
9026:ERROR	CNOS Notification Received for unknown partner <partnerLU>	
	DESCRIPTION	Multiple instances of the CRM may be using the same local LU.

	ACTION	Ensure that multiple configurations running JCRMGW do not use the same local LU.
9027:WARNING	Remote Stop Received for <linkref>	
	DESCRIPTION	The remote host has issued a stop for the specified link.
	ACTION	None. This message for information only.
9028:WARNING	Remote Start Received for <linkref>	
	DESCRIPTION	The remote host has issued a start for the specified link.
	ACTION	None. This message for information only.
9029:ERROR	Undefined Remote LU on link <linkref>	
	DESCRIPTION	The remote LU does not exist as defined.
	ACTION	Check the gateway configuration file and the stack configuration and correct the mis-match.
9030:ERROR	Unable to start session on link <linkref>. Reason=<reason>	
	DESCRIPTION	Link activation failure due to SNA error.
	ACTION	<reason> is the description of the stack return code. Determine the cause and correct.
9031:ERROR	Unable to initialize link <linkref>. Reason=<reason>	
	DESCRIPTION	Link initialization failure due to SNA error.
	ACTION	<reason> is the description of the stack return code. Determine the cause and correct.
9032:ERROR	No Available Session on link <linkref> for context <correlator>	
	DESCRIPTION	Max sessions has been exceeded.
	ACTION	Check session limits in gateway configuration, stack configuration, OLTP application or VTAM. Increase if necessary.

9033:ERROR	Requested Synclevel not supported by link <linkref> for context <correlator> (synclevel <level>)	
	DESCRIPTION	Attempted to issue a request at sync level <level> on a link that does not support that level.
	ACTION	Correct application or gateway configuration.
9035:ERROR	Inbound Request Transform Failed (<status>) for context <correlator>	
	DESCRIPTION	An error has occurred while processing the CICS transform for an inbound DPL request. This normally occurs when the API entry in the gateway configuration for the local service specifies CICS instead of ATMI.
	ACTION	Check gateway configuration for incorrect specification of local service API entry.
9036:ERROR	Inbound Response Transform Failed (<status>) for context <correlator>	
	DESCRIPTION	An error has occurred while processing the CICS transform for an inbound DPL response. This normally occurs when the API entry in the gateway configuration for the local service specifies CICS instead of ATMI.
	ACTION	Check gateway configuration for incorrect specification of local service API entry.
9037:ERROR	Outbound Request Transform Failed (<status>) for context <correlator>	
	DESCRIPTION	An error has occurred while processing the CICS transform for an outbound DPL request. This normally occurs when the API entry in the gateway configuration for the remote service specifies CICS instead of ATMI.
	ACTION	Check gateway configuration for incorrect specification of local service API entry.

9038:ERROR	Outbound Response Transform Failed (<status>) for context <correlator>	
	DESCRIPTION	An error has occurred while processing the CICS transform for an outbound DPL response. This normally occurs when the API entry in the gateway configuration for the remote service specifies CICS instead of ATMI.
	ACTION	Check gateway configuration for incorrect specification of local service API entry.
9040:ERROR	Inbound Confirm not supported	
	DESCRIPTION	Host application is requesting an inbound confirm. This is not supported.
	ACTION	Check host application program and correct.
9041:ERROR	Inbound Confirm for multi-ISRT not supported	
	DESCRIPTION	Host IMS application is requesting an inbound confirm and using multiple ISRT commands. This is not supported.
	ACTION	Check host application program and correct.
9043:ERROR	Missing send last from host (ATMI request/response) for context <correlator>	
	DESCRIPTION	Host application did not issue send last during an outbound request/response service. The host application may have abended.
	ACTION	Check application program and correct.
9044:INFO	DPL program abended with CICS code <abendcode>, program=<progname>	
	DESCRIPTION	The specified host DPL program has abended with the code specified.
	ACTION	None. This message is for information only.
9045:INFO	DPL program failed with CICS rcode <eibrcode>, program=<progname>	

	DESCRIPTION	The specified host DPL program has failed with the `eibrcode` specified.
	ACTION	None. This message is for information only.
9046:ERROR	Invalid combination for Service Context `<correlator>`, `<combination>`	
	DESCRIPTION	The specified `<combination>` is invalid. It will be one of the following: 1. Sync-Level, function, and API 2. Function and API
	ACTION	Examine the gateway configuration and make corrections.
9047:ERROR	Sequence number error for Service Context `<correlator>`, seqno `<seqno>`	
	DESCRIPTION	There has been a sequence number failure for the specified context. Context is out of sequence.
	ACTION	Contact BEA Customer Support.
9048:ERROR	Invalid conversation task for Service Context `<correlator>`, task=`<task>`	
	DESCRIPTION	The conversation has already been terminated.
	ACTION	Contact BEA Customer Support.
9049:ERROR	Invalid task switch for Service Context `<correlator>`, from `<task1>` to `<task2>`	
	DESCRIPTION	An internal protocol violation has occurred.
	ACTION	Contact BEA Customer Support.
9050:ERROR	Transformer creation failed for inbound transaction `<trancode>`	
	DESCRIPTION	An internal error has occurred. Possibly out of memory.
	ACTION	Contact BEA Customer Support.
9051:ERROR	Transformer failed for inbound transaction `<trancode>`	

	DESCRIPTION	An internal error has occurred. Resource name is not present. Mainframe compatibility problem.
	ACTION	Contact BEA Customer Support.
9052:WARNING	Inter-task Message dropped (<verbname>), parm=<parm> From: <task1> to <task2>	
	DESCRIPTION	An internal message between two tasks has been dropped.
	ACTION	None. This message is for information only.
9053:ERROR	Attempt to send <nnnnn> bytes (> 32767)	
	DESCRIPTION	The length of a send request exceeded 32767 (including overhead).
	ACTION	Check application program and correct.
9054:ERROR	Allocation Failure for <trancode> on <remotesysid>: <error>	
	DESCRIPTION	An Allocation error occurred.
	ACTION	The reason for the failure is described by <error>. Correct problem with configuration or application.
9060:WARNING	Inbound Exchange Logs Rejected for <remotesysid>	
	DESCRIPTION	Link not configured for sync level 2.
	ACTION	None. This message is for information only.
9061:WARNING	Link <linkref> not configured for sync level 2	
	DESCRIPTION	Link specified by <linkref> is not configured for sync level 2.
	ACTION	None. This message is for information only.

7 JAM Error Messages

The following table contains a description of error, informational, and warning messages that can be encountered while using the JAM software.

100	**warning: 66 level (RENAMES) is not supported**	
	DESCRIPTION	This language feature is not supported.
	ACTION	No action is necessary.
101	**warning: 88 level (condition name) is not supported**	
	DESCRIPTION	This language feature is not supported.
	ACTION	No action is necessary.
102	**warning: Binary bitfield datatype is not supported**	
	DESCRIPTION	This language feature is not supported.
	ACTION	No action is necessary.
103	**warning: COMP-5 datatype is not supported**	
	DESCRIPTION	This language feature is not supported.
	ACTION	No action is necessary, but it is recommended that the source file be corrected.
104	**warning: COMP-X datatype is not supported**	
	DESCRIPTION	This language feature is not supported.
	ACTION	No action is necessary, but it is recommended that the source file be corrected.
105	**warning: Extraneous '.' ignored**	

	DESCRIPTION	A extra delimiter was encountered, and is ignored.
	ACTION	No action is necessary.
106	**warning: Extraneous OCCURS TO clause, ignored**	
	DESCRIPTION	This clause is not necessary, and is ignored.
	ACTION	No action is necessary, but it is recommended that the source file be corrected.
107	**warning: INDEXED BY clause ignored**	
	DESCRIPTION	This clause is not necessary, and is ignored.
	ACTION	No action is necessary.
108	**warning: Identifier is not unique: {name}**	
	DESCRIPTION	The data item name is not unique, which might cause ambiguity.
	ACTION	No action is necessary.
109	**warning: KEY IS clause ignored**	
	DESCRIPTION	This clause is not necessary, and is ignored.
	ACTION	No action is necessary.
110	**warning: Level number {num} is out of sequence, treating like {num}**	
	DESCRIPTION	The level number of a data item definition does not match previous level numbers, so a default value is assumed.
	ACTION	No action is necessary, but it is recommended that the source file be corrected.
111	**warning: OCCURS lower bound exceeds upper bound ({occurMin} > {occurMax})**	
	DESCRIPTION	The OCCURS ranges are out of order.
	ACTION	No action is necessary, but it is recommended that the source file be corrected.
112	**warning: PICTURE ignored for COMP-1/COMP-2 datatype**	

	DESCRIPTION	The clause is not meaningful for the data item definition.
	ACTION	No action is necessary.
113	**warning: PICTURE ignored for INDEX datatype**	
	DESCRIPTION	The clause is not meaningful for the data item definition.
	ACTION	No action is necessary, but it is recommended that the source file be corrected.
114	**warning: PICTURE ignored for POINTER datatype**	
	DESCRIPTION	The clause is not meaningful for the data item definition.
	ACTION	No action is necessary, but it is recommended that the source file be corrected.
115	**warning: PICTURE ignored for binary bitfield datatype**	
	DESCRIPTION	The clause is not meaningful for the data item definition.
	ACTION	No action is necessary, but it is recommended that the source file be corrected.
118	**warning: Token begins with an unrecognizable character ({char})**	
	DESCRIPTION	An unrecognizable character was encountered in the source file.
	ACTION	No action is necessary, but it is recommended that the source file be corrected.
119	**warning: USAGE ignored for 88-level datatype**	
	DESCRIPTION	The clause is not meaningful for the data item definition.
	ACTION	No action is necessary, but it is recommended that the source file be corrected.
120	**warning: Data item follows a 66-level item**	
	DESCRIPTION	All 66-level items must be the last items within a given group.

	ACTION	Correct the source file.
121	**warning: JUSTIFY clause ignored for non-alphanumeric item**	
	DESCRIPTION	The clause is not meaningful for the data item definition.
	ACTION	No action is necessary, but it is recommended that the source file be corrected.
122	**warning: PICTURE clause ignored for RENAMES datatype**	
	DESCRIPTION	The clause is not meaningful for the data item definition.
	ACTION	No action is necessary, but it is recommended that the source file be corrected.
123	**warning: PICTURE clause required for NO USAGE datatype**	
	DESCRIPTION	The required clause is missing.
	ACTION	Correct the source file.
124	**warning: SIGN clause ignored**	
	DESCRIPTION	The clause is not meaningful for the data item definition.
	ACTION	No action is necessary, but it is recommended that the source file be corrected.
125	**warning: Terminating "." appears to be missing**	
	DESCRIPTION	Data Record definitions must be terminated with a !!
	ACTION	No action is necessary, but it is recommended that the source file be corrected.
126	**warning: USAGE clause ignored for RENAMES item**	
	DESCRIPTION	The clause is not meaningful for the data item definition.
	ACTION	No action is necessary, but it is recommended that the source file be corrected.
127	**warning: OCCURS lower bound assumed to be 1**	

	DESCRIPTION	The lower bound in the OCCURS clause of the DEPENDS ON clause is missing and assumed to be 1.
	ACTION	Items with a DEPENDS ON clause require an OCCURS lower bound.
128	**warning: OCCURS lower and upper bounds should be different**	
	DESCRIPTION	The upper and lower bound in the OCCURS clause of the DEPENDS ON clause are the same.
	ACTION	Items with a DEPENDS ON clause should have different upper and lower bounds.
200	**error: BLANK WHEN ZERO clause ignored for non-zoned item**	
	DESCRIPTION	The clause is not meaningful for the data item definition.
	ACTION	Correct the source file.
201	**error: Bad data item clause**	
	DESCRIPTION	A syntax error or semantic disagreement was encountered while parsing the data item definition.
	ACTION	Correct the source file.
202	**error: Cannot REDEFINE the item: {name}**	
	DESCRIPTION	The name specified is not a valid REDEFINES target.
	ACTION	Correct the source file.
203	**error: Character literal is missing its closing quote**	
	DESCRIPTION	Quoted literals require a closing quote mark.
	ACTION	Correct the source file.
204	**error: Character literal is too long, truncated ({num})**	
	DESCRIPTION	Character literals are truncated beyond a fixed upper limit.
	ACTION	Correct the source file.

205	**error: DEPENDING ON clause requires OCCURS TO upper bound**	
	DESCRIPTION	The required clause is missing.
	ACTION	Correct the source file.
206	**error: DEPENDING ON item is not an integer: {name}**	
	DESCRIPTION	The DEPENDING ON data item is not a numeric integer type.
	ACTION	Correct the source file.
207	**error: DEPENDING ON clause requires an OCCURS clause**	
	DESCRIPTION	The required clause is missing.
	ACTION	Correct the source file.
208	**error: Expected an ASCENDING/DESCENDING KEY IS clause**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
209	**error: Expected BLANK**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
210	**error: Expected a DEPENDING ON clause**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
211	**error: Expected a DEPENDING ON qualified identifier**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.

212	**error: Expected EXTERNAL**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
213	**error: Expected EXTERNAL/GLOBAL**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
214	**error: Expected GLOBAL**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
215	**error: Expected an INDEXED BY clause**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
216	**error: Expected an INDEXED BY qualified identifier**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
217	**error: Expected a JUSTIFIED clause**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
218	**error: Expected a KEY IS qualified identifier**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.

219	**error: Expected LEADING/TRAILING, found '{text}'**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
220	**error: Expected an OCCURS clause**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
221	**error: Expected OCCURS lower bound, found '{text}'**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
222	**error: Expected OCCURS upper bound, found '{text}'**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
223	**error: Expected a PICTURE clause**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
224	**error: Expected a PICTURE specification, found '{text}'**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
225	**error: Expected a REDEFINES clause**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.

226	**error: Expected a REDEFINES identifier, found '{text}'**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
227	**error: Expected a RENAMES THRU identifier, found '{text}'**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
228	**error: Expected a RENAMES clause**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
229	**error: Expected a RENAMES qualified identifier, found '{text}'**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
230	**error: Expected a SIGN clause**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
231	**error: Expected a SYNC clause**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
232	**error: Expected a VALUE clause**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.

233	**error: Expected ZERO**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
234	**error: Expected a ')' following a bitfield size, found '{text}'**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
235	**error: Expected a USAGE data type, found '{text}'**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
236	**error: Expected a bitfield size, found '{text}'**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
237	**error: Expected a data clause, found '{text}'**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
238	**error: Expected a level number, found '{text}'**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
239	**error: Expected an identifier or FILLER, found '{text}'**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.

240	**error: Extraneous BLANK WHEN ZERO clause**	
	DESCRIPTION	The data item definition can only have one such clause.
	ACTION	Correct the source file.
241	**error: Extraneous DEPENDING ON clause**	
	DESCRIPTION	The data item definition can only have one such clause.
	ACTION	Correct the source file.
242	**error: Extraneous EXTERNAL clause**	
	DESCRIPTION	The data item definition can only have one such clause.
	ACTION	Correct the source file.
243	**error: Extraneous GLOBAL clause**	
	DESCRIPTION	The data item definition can only have one such clause.
	ACTION	Correct the source file.
244	**error: Extraneous INDEXED BY clause**	
	DESCRIPTION	The data item definition can only have one such clause.
	ACTION	Correct the source file.
245	**error: Extraneous JUSTIFY clause**	
	DESCRIPTION	The data item definition can only have one such clause.
	ACTION	Correct the source file.
246	**error: Extraneous KEY IS clause**	
	DESCRIPTION	The data item definition can only have one such clause.
	ACTION	Correct the source file.

247	**error: Extraneous OCCURS clause**	
	DESCRIPTION	The data item definition can only have one such clause.
	ACTION	Correct the source file.
248	**error: Extraneous PICTURE clause**	
	DESCRIPTION	The data item definition can only have one such clause.
	ACTION	Correct the source file.
249	**error: Extraneous REDEFINES clause**	
	DESCRIPTION	The data item definition can only have one such clause.
	ACTION	Correct the source file.
250	**error: Extraneous RENAMES clause**	
	DESCRIPTION	The data item definition can only have one such clause.
	ACTION	Correct the source file.
251	**error: Extraneous SIGN clause**	
	DESCRIPTION	The data item definition can only have one such clause.
	ACTION	Correct the source file.
252	**error: Extraneous SYNC clause**	
	DESCRIPTION	The data item definition can only have one such clause.
	ACTION	Correct the source file.
253	**error: Extraneous USAGE clause**	
	DESCRIPTION	The data item definition can only have one such clause.
	ACTION	Correct the source file.

254	**error: Extraneous VALUES clause**	
	DESCRIPTION	The data item definition can only have one such clause.
	ACTION	Correct the source file.
255	**error: Hex string literal must have an even number of digits**	
	DESCRIPTION	Hexadecimal character literals must be composed of an even number of digits.
	ACTION	Correct the source file.
256	**error: INDEXED BY clause requires an OCCURS clause**	
	DESCRIPTION	The required clause is missing.
	ACTION	Correct the source file.
257	**error: Improper bitfield size ({len})**	
	DESCRIPTION	A improper length was specified.
	ACTION	Correct the source file.
258	**error: Improper level-number for REDEFINES item ({levelNo})**	
	DESCRIPTION	The level numbers of redefined data items must match.
	ACTION	Correct the source file.
260	**error: KEY IS clause requires an OCCURS clause**	
	DESCRIPTION	The required clause is missing.
	ACTION	Correct the source file.
261	**error: Level number {n} is out of sequence, treating like {n}**	
	DESCRIPTION	The level number of a data item definition does not match previous level numbers, so a default value is assumed.
	ACTION	Correct the source file.

262	**error: Malformed DEPENDING ON clause**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
263	**error: Malformed INDEXED BY clause**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
264	**error: Malformed KEY IS clause**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
265	**error: Malformed USAGE clause**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
267	**error: Malformed VALUE clause**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
268	**error: Malformed data definition ignored for: {name}**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.
269	**error: Malformed data definition, ignored**	
	DESCRIPTION	A syntax error occurred while parsing the source file.
	ACTION	Correct the source file.

270	**error: Malformed picture specification: '{pic}'**	
	DESCRIPTION	The picture clause contains invalid characters.
	ACTION	Correct the source file.
271	**error: Missing PICTURE clause**	
	DESCRIPTION	The data item definition requires such a clause.
	ACTION	Correct the source file.
272	**error: Missing USAGE and PICTURE clauses**	
	DESCRIPTION	The data item definition requires such a clause.
	ACTION	Correct the source file.
273	**error: Missing VALUE literal constant, found '{text}'**	
	DESCRIPTION	The clause contains a syntax error.
	ACTION	Correct the source file.
274	**error: Missing identifier following IN/OF, found '{text}'**	
	DESCRIPTION	The clause contains a syntax error.
	ACTION	Correct the source file.
275	**error: Missing literal constant after THRU, found '{text}'**	
	DESCRIPTION	The clause contains a syntax error.
	ACTION	Correct the source file.
276	**error: Nonexistent or nonunique DEPENDING ON identifier: {name}**	
	DESCRIPTION	The name specified is ambiguous.
	ACTION	Correct the source file.
277	**error: OCCURS count must be greater than zero ({occurMax})**	
	DESCRIPTION	Arrays must have at least one element.
	ACTION	Correct the source file.

278	**error: REDEFINES identifier cannot be qualified**	
	DESCRIPTION	The clause contains a syntax error.
	ACTION	Correct the source file.
279	**error: REDEFINES item must have the same level number ({levelNo})**	
	DESCRIPTION	The level numbers of redefined data items must match.
	ACTION	Correct the source file.
281	**error: Recovering, skipping to next '.'**	
	DESCRIPTION	A syntax error was encountered, so the rest of the definition is ignored.
	ACTION	Correct the source file.
282	**error: String literal is empty**	
	DESCRIPTION	A quoted literal must contain at least one character.
	ACTION	Correct the source file.
283	**error: USAGE and PICTURE clauses disagree**	
	DESCRIPTION	The clauses specify contradictory types or lengths.
	ACTION	Correct the source file.
284	**error: Word is too long, truncated ({num})**	
	DESCRIPTION	Token words cannot be longer than a certain fixed length.
	ACTION	Correct the source file.
285	**error: PICTURE and SIGN clauses disagree**	
	DESCRIPTION	The clauses specify contradictory types or lengths.
	ACTION	Correct the source file.

300	**Error: An I/O error occurred while generating [{name}]: {error}**	
	DESCRIPTION	Could not write to the output file.
	ACTION	Check the permissions of the output file.
301	**Error: An I/O error occurred while generating [{name}]: {error}**	
	DESCRIPTION	Could not write to the output file.
	ACTION	Check the permissions of the output file.
302	**Error: An I/O error occurred while generating [{name}]: {error}**	
	DESCRIPTION	Could not write to the output file.
	ACTION	Check the permissions of the output file.
303	**Error: An I/O error occurred while generating [{view}]: {error}**	
	DESCRIPTION	Could not write to the output file.
	ACTION	Check the permissions of the output file.
304	**Error: An I/O error occurred while reading the script: {file}**	
	DESCRIPTION	The file could not be read.
	ACTION	Check the permissions of the input file.
305	**Error: EJB specification must contain both a class name and a method name**	
	DESCRIPTION	Proper code cannot be generated without the missing items.
	ACTION	Provide the missing items.
306	**Error: EJB {bean} is not defined.**	
	DESCRIPTION	A nonexistent EJB bean name was referenced.
	ACTION	Specify a different file name.

307	**Error: Parse failed on [{file}].**	
	DESCRIPTION	A syntax error was encountered while parsing the script file.
	ACTION	Correct the script.
308	**Error: The copybook [{file}] was not found.**	
	DESCRIPTION	A nonexistent COBOL source file name was specified.
	ACTION	Correct the misspelling or provide the missing source file.
309	**Error: The script file [{file}] was not found.**	
	DESCRIPTION	A nonexistent file name was specified.
	ACTION	Specify a different file name.
310	**Error: excess method {name} is ignored.**	
	DESCRIPTION	An extraneous method definition was specified.
	ACTION	Remove the duplicate definition.
311	**Error: expecting {token}.**	
	DESCRIPTION	A syntax error occurred while parsing the input file.
	ACTION	Correct the input file.
312	**Error: method {name} is not defined in EJB {bean}.**	
	DESCRIPTION	A nonexistent method was referenced.
	ACTION	Correct the input file.
313	**Error: service {name} is not defined.**	
	DESCRIPTION	A nonexistent service name was referenced.
	ACTION	Correct the input file.

314	**Error: service {service} is not defined.**	
	DESCRIPTION	The service name referenced was not defined.
	ACTION	Provide the missing service name definition or correct the misspelling.
315	**Error: servlet {name} refers to an unknown page ({page}).**	
	DESCRIPTION	A nonexistent page name was referenced.
	ACTION	Correct the input file.

Index