MFS XML Utility Version 9.3.0 User's Guide and Reference

Contents

Chapter 1. Overview of the MFS XML Utility V9.3.0

The MFS XML Utility V9.3.0

The MFS XML Utility is a command line development-time tool that runs on a Microsoft DOS command prompt. This utility generates all of the files needed to web-enable MFS-based IMS transactions. The MFS XML Utility takes MFS source files as input and generates metadata XMI files and Web application archive (WAR) files. Additionally, the utility provides FTP client support so that you can transfer the generated output to an application server such as WebSphere Application Server.

The following topics provide additional information:

- "Prerequisites for the MFS XML Utility Version 9.3" on page 6
- "Parsing MFS source files to generate XMI files"
- "Web.xml file" on page 2
- "Web Application Archive (WAR) files" on page 5
- "User modes"

User modes

You can run the MFS XML Utility in three different modes:

Novice:
The MFS XML Utility prompts you at each input instruction until all instructions are complete. Novice mode is for new users and users that prefer more guidance.

Expert:
You can specify all of the input values as flag parameters in one command.

Batch: You can rerun previously saved flag-value pairs from "Step 1: Generating XMI files from MFS source files" on page 16 or "Step 2: Generating the instance servlet and the web.xml files" on page 20 to generate the XMI and instance servlet files.

For more information, see Chapter 2, "User modes," on page 7.

Parsing MFS source files to generate XMI files

The MFS XML Utility invokes the MFS Importer and uses the Eclipse Modeling Framework (EMF) to serialize each MID/DIF pair, MOD/DOF pair, and MFS TABLE into an XMI file. The XMI files contain all of the application metadata information from the MFS source, including the input and output device descriptors, message descriptors, MID-MOD chaining, device characteristics, and operation semantics. The generated XMI files are then transferred to an XMI repository on WebSphere Application Server and are read for data transformation during runtime.

The optional device characteristics table file specifies the screen size of certain device types. Transfer the file in binary format from MVS to the machine running the MFS XML Utility. You can transfer the MFS source files from MVS in either text or binary format. Because of this, you need to indicate which mode you are using to transfer the files during "Step 1: Generating XMI files from MFS source files" on page 16.

- Files in binary mode are parsed against the host code page. The host code page value is recorded in the XMI file and used by the runtime transformation.
- Files in text mode are parsed against the source code page.

The MFS importer is invoked after you specify the host code page. If the parsing of a specific MFS source file resulted in warnings, the XMI files will still be generated. However, if the parsing resulted in errors,

the XMI files will not be generated and you will be returned to the MFS XML Utility menu. For more information on warnings and errors, see Chapter 5, “MFS XML Utility Messages and Codes,” on page 31. For more information on code pages, see Chapter 6, “Codepages,” on page 47.

The following snippet shows the MID, MOD, DIF, DOF blocks in a MFS source file:

```
IVTNOMI1 MSG TYPE=INPUT,SOR=(IVTNOF,IGNORE),NXT=IVTNO
...
IVTNOMI2 MSG TYPE=INPUT,SOR=(IVTNOF,IGNORE),NXT=IVTNO
...
IVTNO MSG TYPE=OUTPUT,SOR=(IVTNOF,IGNORE),NXT=IVTNOMI1
...
IVTNOF FMT
...
```

Parsing the file shown above, three XMI files will be generated:

```
IVTNOMI1.xmi containing MID (IVTNOMI1) and DIF (IVTNOF) metadata
IVTNOMI2.xmi containing MID (IVTNOMI2) and DIF (IVTNOF) metadata
IVTNO.xmi containing MOD (IVTNO) and DOF (IVTNOF) metadata
```

After the XMI files are parsed, you are prompted to select a device type and feature. The output directory specifies where to save the generated XMI files on the local machine. After the XMI files are generated, move them to a file system folder that is accessible by the WebSphere Application Server manually or using the provided FTP support in the MFS XML Utility. You must specify the file path of the XMI repository, accessed by WebSphere Application Server at runtime, during the run of “Step 2: Generating the instance servlet and the web.xml files” on page 20.

For more information, see “Step 1: Generating XMI files from MFS source files” on page 16.

Web.xml file

The web.xml file contains the following initialization parameters for each of the generated instance servlets:

MFSXMLRepositoryfileURI
: The XMI repository on WebSphere Application Server. It contains the file path from which to load the XMI files during run time.

MFSStyleSheet
: The style sheet URI that is used for rendering the HTML page. See the *MFS Web Enablement Version 9.3.0 User's Guide and Reference* for more information.

serverPlatform
: The platform on which you installed the WebSphere Application Server: Windows- or other systems-based, such as z/OS or AIX.

hostname
: The IMS host name to connect to.

portNumber
: The port number of the IMS host.

dataStore
: The data store of the IMS host.

traceLevel
: The trace level for IMS Connector for Java:
  - Trace level 0: IMS trace level RAS_TRACE OFF for no tracing.
  - Trace level 1: Lists only errors and exceptions.
  - Trace level 2: Adds entry and exit methods.
  - Trace level 3: Prints the contents of buffers sent to and received from IMS Connect.

If trace level is set 1, 2 or 3, the trace output is directly to WebSphere Application Server's trace log file.

executionTimeout (optional)
The IMS resource adapter execution timeout. The execution timeout value for the IMS resource adapter is defined as the maximum amount of time allowed for IMS Connect to send a message to IMS and receive a response from IMS. The execution timeout value is represented in milliseconds and must be a decimal integer in the range of 1 to 3600000. We recommend setting this value to be 5000 milliseconds.

socketTimeout (optional)
The IMS resource adapter socket timeout. The socket timeout is the maximum amount of time IMS Connector for Java will wait for a response from IMS Connect before disconnecting the socket and returning an exception to the client application.

With the socketTimeout property, you can set individual timeout values for a particular interaction using a socket. The value, in milliseconds, can be set on the socketTimeout property in IMSInteractionSpec. If the socketTimeout property is not specified for an interaction or it is set to zero milliseconds, this means there is no socket timeout and the connection will wait indefinitely. The default socket timeout value is zero. We recommend setting this value to be 5000 milliseconds.

userName (optional)
The RACF user name.

Note: During runtime, you can choose to change the RACF information.

password (optional)
The RACF password.

Note: During runtime, you can choose to change the RACF information.

groupName (optional)
The RACF group name.

Note: During runtime, you can choose to change the RACF information.

You can manually modify the parameter values in the web.xml file by extracting the file, making the appropriate updates, and repackaging it back into the WAR file IMS resource adapter socket timeout. Make sure that any updates to the web.xml file are syntactically correct and the structure of the WAR file stays the same.

Recommendation: Use "Step 2: Generating the instance servlet and the web.xml files" on page 20 to generate your web.xml file instead of generating the file manually.

For more information about web.xml files, see "Step 2: Generating the instance servlet and the web.xml files" on page 20.

Sample web.xml file

Here is a sample web.xml file:

```
<!DOCTYPE web-app (View Source for full doctype...)>
- <web-app>
- <servlet>
<servlet-name>KEVINServlet</servlet-name>
<servlet-class>KEVINServlet</servlet-class>
- <init-param>
<param-name>serverPlatform</param-name>
<param-value>1</param-value>
</init-param>
- <init-param>
<param-name>hostName</param-name>
<param-value>ecdvl92.svl.ibm.com</param-value>
</init-param>
- <init-param>
<param-name>dataStore</param-name>
<param-value>IMS1</param-value>
</init-param>
- <init-param>
<param-name>portNumber</param-name>
<param-value>9999</param-value>
</init-param>
- <init-param>
<param-name>MFSXMIRepositoryURI</param-name>
<param-value>file:/c:\xmi</param-value>
</init-param>
- <init-param>
<param-name>MFSStyleSheet</param-name>
<param-value>file:/c:/$Projects/MFSXML/source/exampleIEN6.xsl</param-value>
</init-param>
- <init-param>
<param-name>traceLevel</param-name>
<param-value>0</param-value>
</init-param>
- <init-param>
<param-name>executionTimeout</param-name>
<param-value>5000</param-value>
</init-param>
- <init-param>
<param-name>socketTimeout</param-name>
<param-value>0</param-value>
</init-param>
<param-name>userName</param-name>
<param-value>KEVIN</param-value>
</init-param>
- <init-param>
<param-name>Password</param-name>
<param-value>LO</param-value>
</init-param>
- <init-param>
<param-name>groupName</param-name>
<param-value>KGROUP</param-value>
</init-param>
</servlet>
- <servlet-mapping>
<servlet-name>KEVINServlet</servlet-name>
<url-pattern>/KEVINServlet</url-pattern>
</servlet-mapping>
</web-app>
```

Figure 1. Sample web.xml file

Here is a second sample web.xml file for WAS z/OS (or AIX):

Figure 2. Sample web.xml for WAS z/OS (or AIX)

```
<!DOCTYPE web-app (View Source for full doctype...)>
<web-app>
<!-- zos servlet created on 07/05/2006 11:14:47 PDT-->
  <servlet>
  <servlet-name>zos</servlet-name>
  <servlet-class>zos</servlet-class>
 <init-param>
  <param-name>serverPlatform</param-name>
  <param-value>2</param-value>
 </init-param>
 <init-param>
  <param-name>hostName</param-name>
  <param-value>ecdvl92.svl.ibm.com</param-value>
 </init-param>
 <init-param>
  <param-name>dataStore</param-name>
  <param-value>IMS1</param-value>
 </init-param>
 <init-param>
  <param-name>portNumber</param-name>
  <param-value>9999</param-value>
 </init-param>
 <init-param>
  <param-name>MFSXMIRepositoryURI</param-name>
  <param-value>file:///usr/mfsxml/xmi</param-value>
 </init-param>
 <init-param>
  <param-name>MFSStyleSheet</param-name>
  <param-value>file:///usr/mfsxml/ss/sample3270.xsl</param-value>
 </init-param>
 <init-param>
  <param-name>traceLevel</param-name>
  <param-value>3</param-value>
 </init-param>
 <init-param>
  <param-name>executionTimeout</param-name>
  <param-value>5000</param-value>
 </init-param>
 <init-param>
  <param-name>socketTimeout</param-name>
  <param-value>5000</param-value>
 </init-param>
  </servlet>
  <servlet-mapping>
  <servlet-name>zos</servlet-name>
  <url-pattern>/zos</url-pattern>
  </servlet-mapping>
</web-app>
```

For more information about web.xml files, see “Step 2: Generating the instance servlet and the web.xml files” on page 20.

Web Application Archive (WAR) files

The MFS XML Utility packages the generated instance servlets and deployment descriptor web.xml files into a J2EE compliant WAR file that is deployable on WebSphere Application Server. The deployment web.xml file stores specific style sheet, XMI repository, host connection information, as well as some timeout settings related to IMS resource adapter (IMS Connector for Java). The J2EE-compliant WAR file contains one or more Java files, class files, and web.xml files.

Here is an example of what the J2EE compliant WAR files generated by the MFS XML Utility can contain:

```
/WEB-INF/classes/servlet1.class
/WEB-INF/classes/servlet2.class
/WEB-INF/classes/servlet3.class
/WEB-INF/classes/servlet1.java
/WEB-INF/classes/servlet2.java
/WEB-INF/classes/servlet3.java
.
.
/WEB-INF/web.xml
```

Figure 3. Example of J2EE-compliant WAR file

For more information about WAR files, see "Step 3: Generating the Web Application Archive (WAR) file" on page 22.

Prerequisites for the MFS XML Utility Version 9.3

You must have the following components installed and configured before you can use the MFS XML Utility version 9.3:

- Microsoft DOS on Microsoft Windows

  Note: Set the Microsoft DOS command prompt screen buffer size to 300 x 300.
- IBM MFS Web Enablement Version 9.3
- IBM WebSphere Application Server version 6.1 distributed platforms

Chapter 2. User modes

You can run the MFS XML Utility in one of three different user modes: novice, expert, and batch. These modes are described in:

- "Invoking the MFS XML Utility in novice mode"
- "Invoking the MFS XML Utility in expert mode"
- "Invoking the MFS XML Utility in batch mode" on page 10

Invoking the MFS XML Utility in novice mode

Novice mode is the default user mode.

To invoke the MFS XML Utility in novice mode:

1. From the MFS XML Utility menu, choose selection 1 or 2 and press **Enter** (step 1 is shown below):

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Choose services available to execute from below.
1) Generate XMI files from MFS source files
2) Generate instance servlet and WAS deployment descriptor
3) Generate J2EE compliant WAR (Web application ARchive) file
4) Uploading WAR and XMI files using FTP client
5) Run previously saved batch file
6) Exit
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Please enter your selection here: 1
Step 1: Generate XMI files that represent MID/DIF and MOD/DOF of the MFS source
        This step requires the following information:
        -MFS source files
        -Device Characteristics Table file (Optional)
        -Whether source files are in text or binary format (default to text)
        -Codepage for source files (default to Cp1252)
        -Codepage for host environment (default to Cp037)
        -Device type to format (default to 3270-A02)
        -Device feature to enable (default to ignore)
        -Output directory for generated XMI files (default to installation directory)
```

2. Press **Enter** to accept the default of using novice mode and continue. For more information, see Chapter 3, "Invoking the MFS XML Utility," on page 15.

```
Enter arguments here or press enter to run novice mode. Type '/help' for more information
or 'q' to quit anytime:
```

3. Follow the instructions on each step. Enter `/help` for detailed information. Press `q` to quit and go back to the MFS XML Utility menu.

Invoking the MFS XML Utility in expert mode

Expert mode enables you to specify all of your input values in one instruction.

You can use expert mode only for "Step 1: Generating XMI files from MFS source files" on page 16 or "Step 2: Generating the instance servlet and the web.xml files" on page 20.

To use expert mode, you must know the desired device type and device feature in advance. If you do not know the available device types and features from the parse result of the MFS source files, run in novice mode so you can see the available selections. Otherwise, without specifying the device type and device feature, the MFS XML Utility will select the first device type and feature by alphanumeric order. Each input field is specified by a flag that precedes the value.

The expert mode set of parsing uses the following flag parameters:

Input	Syntax
Device characteristics table	-d or -deviceTableFile
Binary source files (True or False)	-b or -binarySource
Host codepage (see “Host codepage” on page 47 for more information)	-sc or -sourceCodepage
Source codepage (see “Source codepage” on page 48 for more information)	-hc or -hostCodepage
Device type	-dt or deviceType
Device feature	-df or -deviceFeature
Output directory	-o or -outputDirectory
MFS source files	-f or -sourceFile

All of the above parameters are required except for the Device Characteristics Table which is optional. All of the required parameters have a default value except for the MFS source files. You can also see more detailed information by typing `/help` after step 1 is selected in the MFS XML Utility menu. You need to specify `Y` for a binary source file when prompted in step 1 if the source is binary. If you have some binary source files and some non-binary source files, separate parsing of each type is required. The source code page is determined during runtime and is based on what system the MFS XML Utility is running on.

Note: Flag options that have default values can be skipped (for example sourceCodepage, hostcodepage, deviceType, and deviceFeature).

The expert mode uses the following set of servlet flag parameters:

Input	Syntax
Name of the instance Servlet	-n or –instanceServletName
Server Platform	-sp or -serverPlatform
Location of XMI repository on server	-x or –xmiRepository
Name and location of style sheet	-ls or –localStylesheet
Name of the host machine	-ht or –hostname
Port number	-p or –port
IMS name	-i or -ims
RACF username (optional)	-u or –rUserName
RACF group name (optional)	-g or –rGroup
RACF password (optional)	-pw or –rPassword
Trace level for IMS Connector for Java (optional)	-t or -traceLevel
Execution timeout for IMS Connector for Java (optional)	-e or -executionTimeout
Socket timeout for IMS Connector for Java (optional)	-s or -socketTimeout

To invoke the MFS XML Utility in expert mode:

1. From the MFS XML Utility window, choose selection 1 or 2 and press **Enter**. The example shows choosing step 1:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Choose services available to execute from below.
1) Generate XMI files from MFS source files
2) Generate instance servlet and WAS deployment descriptor
3) Generate J2EE compliant WAR (Web application ARchive) file
4) Uploading WAR and XMI files using FTP client
```

```
5) Run previously saved batch file
6) Exit
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Please enter your selection here: 1
Step 1: Generate XMI files that represent MID/DIF and MOD/DOF of the MFS source
        This step requires the following information:
        -MFS source files
        -Device Characteristics Table file (Optional)
        -Whether source files are in text or binary format (default to text)
        -Codepage for source files (default to Cp1252)
        -Codepage for host environment (default to Cp037)
        -Device type to format (default to 3270-A02)
        -Device feature to enable (default to ignore)
        -Output directory for generated XMI files (default to installation directory)
```

2. To run in expert mode, you must enter arguments. Type `/help` and press **Enter** to see the list of expert mode options:

```
Enter arguments here or press enter to run novice mode. Type '/help' for more
information or 'q' to quit anytime:
>>/help
Expert mode importer: [-options]
where required options include:
-f -sourceFile where (-sourceFile file) or (-f file) can be repeated and
   where file can be a directory containing multiple MFS source fileswhere
   optional options include:
-d -deviceTableFile name and location of device characteristics table
-b -binarySource "True" if source file in binary format, "False" otherwise.
   Default is "False"
-sc -sourceCodepage Codepage encoding of MFS source files.
    Default set to codepage of local Java environment
-hc -hostCodepage Codepage encoding on MFS host. Default set to Cp037
-dt -deviceType Target device type.
    Default device type is the first alphabetically sorted device type
-df -deviceFeature Target device feature.
    Default device feature is the first alphabetically sorted device feature
-o -outputDirectory Target location for generated XMI and servlet files.
   Defaults to current directory

Expert mode servlet generator: [-options]
where required options include:
-n -instanceServletName Name of this instance servlet
-x -xmiRepository Location of XMI repository on web server
-ls -localStylesheet Name and location of stylesheet
-ht -hostname name of MFS host machine
-p -port port number.
-i -ims IMS name
-v -imsVerb verb for IMS Connector. Default set to SYNC_SEND_RECEIVE
-t -traceLevel Trace level of IMS Connector for Java (IC4J)
   where optional options include:
-u -rUserName RACF username
-g -rGroup RACF group name (required if RACF username is provided)
-pw -rPassword RACF password (required if RACF username is provided)
-e -executionTimeout IMS Connector for Java execution timeout value (in milliseconds).
   If not specified, global value from IMS Connect will be used.
-s -socketTimeout IMS Connector for Java socket timeout value (in milliseconds).
   Default value is 0.
Press enter to run novice mode, otherwise enter arguments here to run in
expert or batch mode or type /help:
```

3. Specify your flag options and press **Enter** (the step 1 parameters are shown):

```
Enter arguments here or press enter to run novice mode. Type '/help' for more
information or 'q' to quit anytime:
>>-sourceFile bcust1.mfs -outputDirectory MASDBULO\ -sourceCodepage MS950
-hostCodepage Cp037 -deviceType 3270,2 -deviceFeature Ignore
Parsing files...
```

```
parsing bcust1.mfs

Writing to C:\MFSXMLUtility\device_types.log completed
```

or the equivalent:

```
Enter arguments here or press enter to run novice mode. Type '/help' for more
information or 'q' to quit anytime:
>>-f bcust1.mfs -o MASDBULO\ -sc MS950 -hc Cp037 -dt 3270,2 -df Ignore
Parsing files...
parsing bcust1.mfs

Writing to C:\MFSXMLUtility\device_types.log completed
```

Note: You can add the prefix `-f` or `-sourceFile` to each MFS source file name and specify multiple source files:

```
-f dfsivf1.mfs -f bcust1.mfs
```

or the equivalent:

```
-sourceFile dfsivf1.mfs -sourceFile bcust1.mfs
```

or `-f` for file directory to parse all the source files at once, like `-f c:\mfs`.

4. Specify yes (`y`) or no (`n`) if you would like to see the parse output and press **Enter** (default is `n`):

```
Parse successfully. Would you like to see the parse output? (y|n; default is no):

Parse output log will be created.


Writing to C:\MFSXMLUtility\parse.log completed


The following XMI files were generated:
 C:\MFSXMLUtility\IVTNOMI1.xmi
 C:\MFSXMLUtility\IVTNO.xmi
```

Invoking the MFS XML Utility in batch mode

Batch mode provides a convenient way to re-run the same job. At the end of "Step 1: Generating XMI files from MFS source files" on page 16 or "Step 2: Generating the instance servlet and the web.xml files" on page 20 in the novice or expert mode, you are prompted to save the current parameters into a batch parameter file. If you choose yes, a batch file is created that allows you to run the same job later without specifying the same arguments all over again.

The name of the batch file from step 1 is created using the last message descriptor name in the last parsed MFS source file. The name of the batch file from step 2 is created by combining the name of the instance servlet and "_step2.txt".

You can invoke the MFS XML Utility in batch mode using two methods. The first method allows you to enter batch mode and specify all of your commands directly from a Microsoft DOS command prompt. The second method allows you to enter batch mode from within the MFS XML Utility interface using "(Optional) Step 5: Running a batch file" on page 27.

Method 1

To invoke the MFS XML Utility in batch mode:

1. From within a Microsoft DOS command prompt, navigate to your `MFSXMLUtility` directory and type `mfsxml -batch <batchfilename>`, where *batchfilename* is the absolute or relative batch parameter filename (example shows the absolute file path):

   ```
   C:\MFSXMLUtility>mfsxml -batch C:\MFSXMLUtility\PHONEBOOK\IVTNOMI1_step1.txt
   ```

2. The MFS XML Utility checks for the required classpath JAR files:

```
CHECKING FOR CLASSPATH JAR FILES
  .........
  checking common.jar...
  checking ecore.jar...
  checking ecore.xmi.jar...
  checking j2ee.jar...
  checking MFSUtility.jar...
  checking MFSTDTDLang.jar...
  checking MFSImporter.jar...
  checking MFSRuntime.jar...
  checking Java Version...
  java version "1.4.2_04"
```

3. Verify that the step 1 commands appear before the step 2 commands in your batch file by indicating yes (y) or no (n) and press **Enter**

   Important: It is required that all of the step 1 arguments are placed before step 2 arguments in the batch file.

   :

```
Note that all the step 1 batch commands need to be placed BEFORE step 2 batch commands in
the batch file
Continue? (y|n; default is yes)

RUNNING BATCH FILE C:\MFSXMLUtility\PHONEBOOK\IVTNOMI1_step1.txt

Beginning Step 1: Generating XMI files

Parsing files...
parsing dfsivf1.mfs

Writing to C:\MFSXMLUtility\device_types.log completed
```

4. Specify whether or not (yes (y) or no (n)) you would like to see the parse output, and press **Enter**:

```
Parse successfully. Would you like to see the parse output? (y|n; default is no): n

Parse output log will be created.


Writing to C:\MFSXMLUtility\parse.log completed


Utility completed.
************************************************************
The following XMI files were generated:
 C:\MFSXMLUtility\PHONEBOOK\IVTNO.xmi
 C:\MFSXMLUtility\PHONEBOOK\IVTNOMI1.xmi
************************************************************

C:\MFSXMLUtility>
```

Method 2

To invoke the MFS XML Utility in batch mode:

1. From the MFS XML Utility window, choose selection 5 and press **Enter**:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Choose services available to execute from below.
1) Generate XMI files from MFS source files
2) Generate instance servlet and WAS deployment descriptor
3) Generate J2EE compliant WAR (Web application ARchive) file
4) Uploading WAR and XMI files using FTP client
5) Run previously saved batch file
6) Exit
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Please enter your selection here: 5
```

2. Specify your batch files, separated by a space, and press **Enter** (filenames can also be specified in an absolute or relative path).

   The batch file in this example (PhoneBook.txt) contains batch commands for step 1 and step 2. The invocation of this batch file parses the MFS source file, generates the XMI files, and prompts you about whether or not to generate a WAR file that contains both the web.xml file and the instance servlet class file.

   ```
   Enter batch files separated by space: .\PHONEBOOK\PhoneBook.txt
   RUNNING BATCH FILE .\PHONEBOOK\PhoneBook.txt

   Beginning Step 1: Generating XMI files
   Starting Step 2: Generate and compile instance servlet(s)
   Generating servlet complete
   Servlet is being compiled...
   Compilation completed. Generating PhoneBookServlet.class

   Moving generated instance servlet class file to WEB-INF directory...
   Starts to put files in the WEB-INF directory...
   Servlet is being compiled...
   Compilation completed. Generating PhoneBookServlet.class.

   Generating servlet deployment descriptor....
   Instance servlet deployment descriptor files created.
   ```

3. Indicate if you would like to generate a WAR file by specifying `y` (yes) or `n` (no) and pressing **Enter** (default is `y`):

   ```
   Would you like to generate a WAR file now? (Y|N; default is n):y
   ```

4. Enter the name of the WAR file and press **Enter**:

   ```
   Enter the name of the WAR file: demo2
   ```

5. Indicate if you want to package additional files with this WAR file by specifying `y` (yes) or `n` (no) and pressing **Enter**:

   ```
   Do you want to package additional files such as pictures with this WAR file?
   (y|n; default is no)n
   added manifest
   adding: WEB-INF/(in = 0) (out= 0)(stored 0%)
   adding: WEB-INF/classes/(in = 0) (out= 0)(stored 0%)
   adding: WEB-INF/classes/PhoneBookServlet.class(in = 379) (out= 268)(deflated 29%)
   adding: WEB-INF/classes/PhoneBookServlet.java(in = 553) (out= 348)(deflated 37%)
   adding: WEB-INF/web.xml(in = 987) (out= 315)(deflated 68%)
   WAR file generated.
   ```

Example: creating a batch file from multiple step 1 and step 2 parameters

The following is a batch file created from multiple step 1 and step 2 parameters. The batch file parses multiple MFS source files and generates three instance servlets and a web.xml file. It is equivalent to one step 1 run and three step 2 runs. It contains all of the arguments for generating both an XMI file and a WAR file. When you create a batch file manually, place all of the step 1 parameters before the step 2 parameters.

Note: The following example of a batch file is shown in multiple lines for illustration purposes only. Each run of the step 1 arguments need to be on one line followed by each run of the step 2 arguments on another line.

```
-sourceFile C:\ImporterTestCases_8_23_04\customer_mfs\DEOBMINQ.MFS
-sourceFile C:\ImporterTestCases_8_23_04\customer_mfs\deoddnd.mfs
-sourceFile C:\ImporterTestCases_8_23_04\customer_mfs\DFSDBCST.mfs
-sourceFile C:\ImporterTestCases_8_23_04\customer_mfs\dfsivf1.mfs
-sourceFile C:\ImporterTestCases_8_23_04\customer_mfs\IGL.EDVR.PROD060.MFS
-sourceFile C:\ImporterTestCases_8_23_04\customer_mfs\MASDBULO.MFS
-sourceFile C:\ImporterTestCases_8_23_04\customer_mfs\OE4COR01.mfs
```

```
-sourceFile C:\ImporterTestCases_8_23_04\customer_mfs\OE4MSGO1.MFS
-sourceFile C:\ImporterTestCases_8_23_04\customer_mfs\TTT3D.MFS
-outputDirectory C:\MFSXMLUtility\Multiple\
-sourceCodepage Cp1252
-hostCodepage Cp037
-deviceType 3270,2
-deviceFeature Ignore

-instanceServletName fancyterminal31
-outputDirectory C:\MFSXMLUtility\fancyterminal31\
-serverPlatform 2
-xmiRepository file:///usr/mfs/xmi
-localStylesheet file:///usr/mfs/ss/sampleWeb.xsl
-hostname ecdb31.svl.ibm.com -port 9999 -ims IMS1
-traceLevel 3
-executionTimeout 5000 -socketTimeout 6000

-instanceServletName fancyterminal91
-outputDirectory C:\MFSXMLUtility\fancyterminal91\
-serverPlatform 1
-xmiRepository file:/c:\xmi
-localStylesheet file:/c:\$Projects\MFSXML\source\sampleWeb.xsl
-hostname ecdvl91.svl.ibm.com -port 9999 -ims IMS1
-traceLevel 3
-executionTimeout 6000 -socketTimeout 6000

-instanceServletName fancyterminal92
-outputDirectory C:\MFSXMLUtility\fancyterminal92\
-serverPlatform 1
-xmiRepository file:/c:\xmi
-localStylesheet file:/c:\$Projects\MFSXML\source\sampleWeb.xsl
-hostname ecdvl92.svl.ibm.com -port 9999 -ims IMS1
-traceLevel 3
-executionTimeout 5000 -socketTimeout 0
```

Note: It is not recommended that you manually modify the values in the batch file other than when you combine several batch files into a single batch file.

Chapter 3. Invoking the MFS XML Utility

The MFS XML Utility is invoked by starting the mfsxml.bat file using a Microsoft DOS command prompt.

To invoke the MFS XML Utility:

1. From a Microsoft DOS command prompt type `mfsxml` and press **Enter**:
   The MFS XML Utility checks to see if the following JAR files are present:

Name	Description
MFSRuntime.jar	MFS Web Enablement runtime classes
MFSTDTDLang.jar	MFS Web Enablement runtime classes
MFSImporter.jar	MFS Importer for parsing MFS source files
common.jar	EMF (Eclipse Modeling Framework) core classes
ecore.jar	EMF (Eclipse Modeling Framework) core classes
ecore.xmi.jar	EMF (Eclipse Modeling Framework) core classes
j2ee.jar	J2EE classes used to compile instance servlets
MFSUtility.jar	MFS XML Utility classes

   The common.jar, ecore.jar, ecore.xmi.jar, and j2ee.jar files are obtained from the WebSphere Application Server library directory.

   The example below shows the initial check of required JAR files and JDK:

   Note: The MFS XML Utility stops running if one of the required jar files is missing

```
C:\MFSXMLUtility>mfsxml
CHECKING FOR CLASSPATH JAR FILES
  .........
  checking common.jar...
  checking ecore.jar...
  checking ecore.xmi.jar...
  checking j2ee.jar...
  checking MFSUtility.jar...
  checking MFSTDTDLang.jar...
  checking MFSImporter.jar...
  checking MFSRuntime.jar...
  checking Java Version...
  java version "1.4.2_04"
```

Figure 4. MFS XML Utility checking for required JAR files and JDK

2. A screen displays an overview of the steps that are involved with the MFS XML Utility along with a description of each step:

```
Welcome to the MFS XML Utility Tool, a development time tool for the MFS Web Enablement!

This utility creates web-enabled MFS applications based on existing/working MFS
source files in the following steps.
Step 1: Generate XMI files from MFS source files
        *Generate XMI files that represent MID/DIF and MOD/DOF of the MFS source.
        *This is accomplished by invoking the MFS Importer, which parses the MFS source
         file for a particular MFS application,
         to generate three kinds of XMI files to represent MID/DIF, MOD/DOF, and MFS table
         if necessary.
        *The XMI files represents all the application interface information encapsulated by the
         MFS source including the input and output messages, display information, MFS flow
```

```
         control, device characteristics and operation semantics.

Step 2: Generate instance servlet and WAS deployment descriptor
        *Generate and compile instance servlet to create .class files used during runtime by
         the backend MFS application.
        *Generate web.xml deployment descriptors used by WebSphere Application Server.

Step 3: Generate J2EE compliant WAR (Web application ARchive) file
        *Generate WAR (Web Application aRchive) file packaging one or more instance servlet
         class files generated in Step 2.
        *The content and structure of the WAR file tells WAS what/how/where to deploy the
         generated instance servlets.

Step 4: Upload genereated WAR and XMI files using FTP client
        *This step allows user to upload files created in Step 1, 2, and 3 to host where
         WebSphere Application Server is running.
         After the FTP operation, WAS administrator can then deploy the WAR file manually
         using the WAS admin console.
        *The uploaded XMI files should be moved to where XMI repository is (specified in
         instance servlet) for use at runtime to generate HTML.

Step 5: Run previously saved batch file

Step 1 & 2 can be run in the following modes:
        *Novice mode - provides step-by-step instructions for input values.
        *Expert mode - reads flag-value pairs to quickly create web enabled MFS applications.
        *Batch mode - reads flag-value pairs from a file to generate XMI and servlet files.
```

3. You are prompted to choose from a menu of services:

```
Choose services available to execute from below.
1) Generate XMI files from MFS source files
2) Generate instance servlet and WAS deployment descriptor
3) Generate J2EE compliant WAR (Web application ARchive) file
4) Uploading WAR and XMI files using FTP client
5) Run previously saved batch file
6) Exit

Please enter your selection here:
```

For more information about each of the available services, see:

- "Step 1: Generating XMI files from MFS source files"
- "Step 2: Generating the instance servlet and the web.xml files" on page 20
- "Step 3: Generating the Web Application Archive (WAR) file" on page 22
- "(Optional) Step 4: Uploading WAR and XMI files using an FTP client" on page 26
- "(Optional) Step 5: Running a batch file" on page 27

Step 1: Generating XMI files from MFS source files

In step 1, you generate XMI files from MFS source files.

To generate XMI files from the MFS source files:

1. From the MFS XML Utility window, choose selection 1 and press **Enter**:

```
Choose services available to execute from below.
1) Generate XMI files from MFS source files
2) Generate instance servlet and WAS deployment descriptor
3) Generate J2EE compliant WAR (Web application ARchive) file
4) Uploading WAR and XMI files using FTP client
5) Run previously saved batch file
6) Exit
```

```
Please enter your selection here: 1
Step 1: Generate XMI files that represent MID/DIF and MOD/DOF of the MFS source
        This step requires the following information:
        -MFS source files
        -Device Characteristics Table file (Optional)
        -Whether source files are in text or binary format (default to text)
        -Codepage for source files (default to Cp1252)
        -Codepage for host environment (default to Cp037)
        -Device type to format (default to 3270-A02)
        -Device feature to enable (default to ignore)
        -Output directory for generated XMI files (default to installation directory)
```

2. Enter arguments to run in expert mode or press **Enter** to run in novice mode. For more information about novice mode or expert mode, see "Invoking the MFS XML Utility in novice mode" on page 7 or "Invoking the MFS XML Utility in expert mode" on page 7.

```
Enter arguments here or press enter to run novice mode. Type '/help' for more
information or 'q' to quit anytime:
>>

Beginning Step 1: Generating XMI files...
```

3. Specify the MFS source files or directory that contains the MFS source files, and press **Enter**:
   - Here's an example of specifying a source file:

     ```
     Specify MFS source files or directory containing MFS source files: c:\MFSXMLUtility\dfsivf1.mfs
     You selected c:\MFSXMLUtility\dfsivf1.mfs
     ```

   - Here's an example of specifying a directory:

     ```
     Specify MFS source files or directory containing MFS source files: C:\MFS
     Added C:\MFS\DEOBMINQ.MFS
     Added C:\MFS\deoddnd.mfs
     Added C:\MFS\DFSDBCST.mfs
     Added C:\MFS\dfsivf1.mfs
     Added C:\MFS\IGL.EDVR.PROD060.MFS
     Added C:\MFS\MASDBULO.MFS
     Added C:\MFS\OE4COR01.mfs
     Added C:\MFS\OE4MSG01.MFS
     Added C:\MFS\TTT3D.MFS
     You selected C:\MFS
     ```

   - Here is an example of specifying more source files by separating the file names with a space. Specify MFS source files or directory containing MFS source files:

     ```
     c:\MFSXMLUtility\dfsivf1.mfs c:\MFSXMLUtility\oe4cnio1x.mfs
     ```

   Note: When parsing MFS source files that contain a MFS COPY statement, ensure that the copy source file is in the same directory as the MFS source file. Do not specify the copy source file itself, specify only the MFS source file that contains the MFS COPY statement.

4. (Optional) Specify the device characteristics table and press **Enter** (default is `none`):

```
Specify device characteristics table (Optional):
No device characteristics table selected
```

5. Indicate if the MFS source files are in binary mode by typing `y` (yes) or `n` (no) and then pressing **Enter** (default is `n`):

```
Is the source in binary mode (y/n; default is n):
>> You entered n by default
```

6. Specify the source codepage and press **Enter** (default is `Cp1252` for Windows Latin 1). For more information, see "Source codepage" on page 48.

```
Specify codepage for source (your system default is Cp1252):
>> You entered Cp1252 by default.
```

7. Specify the host codepage for the MFS source and press **Enter** (default will be based on system locale). For more information, see "Host codepage" on page 47.

```
Specify codepage for host (Default is Cp037):
>> You entered Cp037 by default.
```

8. Specify the output directory and press **Enter** (default is the MFS XML Utility installation directory or the output directory that was specified during last run):

```
Specify output directory (Default is C:\MFSXMLUtility\): PHONEBOOK
>> You entered C:\MFSXMLUtility\PHONEBOOK\
Parsing files...
```

- The following example shows the MFS Importer catching parser warnings:

```
Parse failed/warnings. Would you like to see the parse output? (y|n; default is yes):
Parse succeeded with warnings

C:\ImporterTestCases\xref3.mfs Returned a warning message:

IXFI005W: An error occurred while loading an external reference tables\TBL2.xmi

IXFI006W: The parser generated default MFSTable to resolve relationship TBL2


Press enter to continue...
```

  These warnings occurred because in step 1 the MFS Importer is trying to load `TBL2.xmi` based on the following in the MFS source:

```
TABLE2 DFLD POS=(15,17),LTH=9,OPCTL=TBL2
DFLD 'TABLE3: ',POS=(17,2)

TABLE3 DFLD POS=(17,17),LTH=9,OPCTL=TBL2

DFLD 'INPUT',POS=(21,2)
```

- Here is an example of the MFS importer catching an error:

```
Parse failed/warnings. Would you like to see the parse output? (y|n; default is yes):
Parse failed

C:\ImporterTestCases_8_23_04\error1.mfs Returned a parse

error: com.ibm.etools.mfs.importer.ParseException:

Encountered "5" at line 7, column 24.

Was expecting one of:

"(" ...

"(" ...
```

  Based on the MFS source file along with that error message explanation, the following error was found in the MFS source file:

```
DPAGE CURSOR=(5,20)
```

  which instead should be:

```
DPAGE CURSOR=((5,20))
```

  This is why the MFS Importer complains about finding “5” where it should be finding “(” on column 24.

9. Choose a device type and press **Enter** (default is the first device type in a device type list sorted in alphabetic order):

```
Choose one of the following device types (Default is 3270-A02; '?' for help):
1)3270-A02
=>
You selected 3270-A02
```

Note: The device types in the MFS source files will be listed. If a large amount of MFS source files are specified, all of the device types in the first five MFS source files will be listed, instead of all of the device types. The chosen device type from the list will then be used in the serialized XMI file. For any MFS source file that has only one device type, if you do not choose that device type from the list, then the device type found in the original MFS source will be serialized into the XMI file. For any MFS source file that has more than one device type, and the device type chosen from the list is not one of the device types found in the original MFS source, then the last device type listed in the MFS source will be serialized into the XMI file. For more information on device types, see "Invoking the MFS XML Utility in expert mode" on page 7.

10. Choose a device feature and press **Enter**.

    Note: The device type default is chosen based on the first device type found in the device type list which is sorted in alphanumeric order. For example, if an MFS source file contains the following 4 device types, this is the order as they will appear in the MFS source file:

    ```
    3270-A2
    3270, 2
    3270-A04
    3270-A02
    ```

    The system will sort the above device types based on alphanumeric order. The sorted list will look like the following:

    ```
    3270,2
    3270-A02
    3270-A04
    3270-A2
    Choose one of the following device features (Default is Ignore):
    1)Ignore
    =>
    You selected Ignore
    parsing c:\MFSXMLUtility\dfsivf1.mfs

    Writing to C:\MFSXMLUtility\device_types.log completed
    ```

    The default device type is the first one displayed, in this example 3270,2, since it is the first on the list.

11. Specify yes (y) or no (n) if you would like to see the parse output and press **Enter** (default is n):

    ```
    Parse successfully. Would you like to see the parse output? (y|n; default is no):

    Parse output log will be created.


    Writing to C:\MFSXMLUtility\parse.log completed


    The following XMI files were generated:
     C:\MFSXMLUtility\PHONEBOOK\IVTNOMI1.xmi
     C:\MFSXMLUtility\PHONEBOOK\IVTNO.xmi
    ```

12. If you want to save your input values for later execution in batch mode, type y and press **Enter**. If you do not want to save your input values to a batch file, press **Enter** to accept the default of n:

    ```
    Do you wish to save your input values to a batch file? (y|n ; default is no)y
    Writing batch file to C:\MFSXMLUtility\PHONEBOOK\IVTNOMI1_step1.txt...
    Step 1 batch file created

    The following batch file was generated:
     C:\MFSXMLUtility\PHONEBOOK\IVTNOMI1_step1.txt

    Step 1 completed.
    ```

Step 2: Generating the instance servlet and the web.xml files

In step 2, you generate the instance servlet class files and web.xml files for packaging into a Web Application Archive (WAR) file in step 3.

To generate the instance servlet and web.xml files:

1. From the MFS XML Utility window, choose selection 2 and press **Enter**:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Choose services available to execute from below.
1) Generate XMI files from MFS source files
2) Generate instance servlet and WAS deployment descriptor
3) Generate J2EE compliant WAR (Web application ARchive) file
4) Uploading WAR and XMI files using FTP client
5) Run previously saved batch file
6) Exit
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Please enter your selection here: 2

Step 2: Generate and compile instance servlet used during runtime for the backend MFS
        application;
        This step requires the following information:
        -Name of the this instance servlet
        -Location of XMI repository on web server (default to last value)
        -Name and location of stylingsheet on local machine to copy to web server (default to last
         value)
        -Host name or IP address of IMS (default to last value)
        -Port number of host (default to last value)
        -IMS datastore name (default to last value)
        -RACF username (optional)
        -RACF group (optional)
        -RACF password (required if RACF username is specified)
        -Trace Level of IMS Connect for Java (default to 1)
        -Execution timeout value in milliseconds (global level timeout value will be used if
         not specified)
        -Socket timout value in milliseconds (default the connection will wait indefinitely if
         not set)

Begin Servlet Generation....
```

2. Press **Enter** to generate the servlet in novice mode. For more information, see "Invoking the MFS XML Utility in novice mode" on page 7. To generate the servlet in expert mode, enter the servlet arguments and press **Enter**. For more information, see "Invoking the MFS XML Utility in expert mode" on page 7. This example uses novice mode:

```
Press Enter to generate servlet in novice mode, otherwise enter servlet arguments for expert mode
or type '/help' or 'q':
>>
```

3. Enter the name of the instance servlet and press **Enter**:

```
Please enter the name of this instance servlet:PHONEBOOK
```

4. Specify the output directory for your instance servlet and press **Enter** (the last output directory specified or the MFS XML Utility installation directory):

```
Specify output directory (default is C:\MFSXMLUtility\PHONEBOOK\):
You have selected an existing directory! Files with the same name will be over-written
without warnings!
Continue? (y|n; default is yes)
>> You entered C:\MFSXMLUtility\PHONEBOOK\
```

5. Specify the number 1 or 2 to indicate the operating system and press **Enter**

```
Please select the platform where WebSphere Application Server is located:
1) WINDOWS
2) Other Systems (for example, z/OS and AIX)
>> 1
You chose the WINDOWS platform.
```

6. Specify the file path URI of the XMI repository on WebSphere Application Server and press **Enter** (default is the last XMI repository that specified or the MFS XML Utility installation directory):

```
Specify target location of XMI repository on web server ('?' for help): c:\xmi
>> You entered file:/c:\xmi
```

7. Specify the target location of your style sheet on WebSphere Application Server and press **Enter** (default is the last style sheet file specified)

```
Specify location of styling sheet ('?' for help): c:\$Projects\MFSXML\source\sample3270.xsl
>> You entered file:/c:\$Projects\MFSXML\source\sample3270.xsl
```

8. Specify the IMS hostname or IP address and press **Enter** (default is the last host name specified):

```
Specify IMS hostname or IP address ('?' for help): ecdb31.svl.ibm.com
>> You entered ecdb31.svl.ibm.com
```

9. Specify a host port number and press **Enter** (default is the last host port number specified):

```
Specify a port number ('?' for help): 9999
>> You entered 9999
```

10. Specify the IMS datastore name and press **Enter** (default is the last IMS datastore name specified):

```
Specify IMS datastore name ('?' for help): IMS1
>> You entered IMS1
```

11. (Optional) You can specify RACF information, or you can skip this step by pressing **Enter** (default is to skip this step). The RACF that you enter here is loaded as the initial value during runtime if information is not specified during runtime.

```
Note that the following RACF information will be loaded as the initial RACF
specification if information is specified during runtime.
Specify RACF user name (Optional; '?' for help):
No value entered
```

Note: The password is masked on the DOS screen. You can verify or update the value in the generated web.xml file.

12. Specify the trace level (from 0 to 3) for IMS Connector for Java and press **Enter** (default is 1)

```
Specify trace level for IMS Connector for Java from 0 to 3 (default is 0; '?' for help): 3
>> You entered trace level 3
```

13. The instance servlet is generated and compiled in the output directory that is specified:

```
Generating servlet......completed
Servlet is being compiled.......completed.
```

14. Specify the execution timeout value (in milliseconds) for IMS Connector for Java and press **Enter**:

```
Specify execution timeout value in milliseconds for IMS Connector for Java ('?' for help): 5000
>> You entered execution timeout value to be 5000 milliseconds
```

15. Specify the socket timeout value (in milliseconds) for IMS Connector for Java and press **Enter** (default is 0):

```
Specify socket timeout value in milliseconds for IMS Connector for Java (default is 0; '?'
for help): 3000
>> You entered socket timeout value to be 3000 milliseconds.
```

16. If you want to save your input values for later execution in batch mode, type `y` and press **Enter**. If you do not want to save your input values to a batch file, press **Enter** to accept the default of `n`:

```
Do you wish to save your input values to a batch file (RACF information will NOT be saved)?
(y|n ; Default is no)y
Writing batch file to C:\MFSXMLUtility\PHONEBOOK\PHONEBOOK_step2.txt...
Batch file created
```

17. The deployment descriptor and web.xml files are generated:

```
Generating servlet deployment descriptor......generated.

Starts to put files in the WEB-INF directory...
Compile servlet to be packaged into WAR file....
Servlet is being compiled.......completed.

The following servlet file was generated:
```

```
 C:\MFSXMLUtility\PHONEBOOK\PHONEBOOK.java
The following servlet class file was generated:
 C:\MFSXMLUtility\PHONEBOOK\PHONEBOOK.class
The following batch file was generated:
 C:\MFSXMLUtility\PHONEBOOK\PHONEBOOK_step2.txt
The following segment of web.xml file was generated:
 C:\MFSXMLUtility\PHONEBOOK\PHONEBOOKWeb.xml
Step 2 completed.
```

Note: The servlet is compiled a second time in the example above for backup purposes.

Refer to the sample web.xml file in "Web.xml file" on page 2 for information about a single servlet.

Step 3: Generating the Web Application Archive (WAR) file

In step 3, the MFS XML Utility generates a Web Application Archive (WAR) file from one or more instance servlets and the web.xml file that you generated in step 2.

To generate the WAR file:

1. From the MFS XML Utility window, choose selection 3 and press **Enter**:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Choose services available to execute from below.
1) Generate XMI files from MFS source files
2) Generate instance servlet and WAS deployment descriptor
3) Generate J2EE compliant WAR (Web application ARchive) file
4) Uploading WAR and XMI files using FTP client
5) Run previously saved batch file
6) Exit
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter your selection here: 3
Step 3: Generate WAR (Web Application aRchive) file containing one or more instance servlets
        In order to generate WAR file, you must first complete step 2.
        This step requires the following information:
        -Previously generated instance servlet class file(s) in the WEB-INF\classes directory
        -Previously generated deployment descriptor (web.xml) in the WEB-INF directory

        *Examine the content of the web.xml file in C:\MFSXMLUtility\WEB-INF\ and make
         any necessary additions.*
        *Note that the only web.xml file that will be packaged into the WAR file is in
         C:\MFSXMLUtility\WEB-INF\
     The other generated web.xml segment is used for reference purposes only

This WAR file will be generated with the following instance servlets.
1) .\WEB-INF\classes\PHONEBOOK.class
2) .\WEB-INF\classes\PHONEBOOK.java
```

2. Enter the name of your WAR file and press **Enter**:

```
Enter the name of this WAR file: PB
```

3. Indicate if you would like to include additional files in your WAR file, for example GIF or JPG files, (default is no) and press **Enter**:

```
Do you want to package additional files such as pictures with this WAR file? (y|n; default is no)
```

4. The WAR file is generated:

```
added manifest
adding: WEB-INF/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/PHONEBOOK.class(in = 379) (out= 270)(deflated 28%)
adding: WEB-INF/classes/PHONEBOOK.java(in = 614) (out= 404)(deflated 34%)
adding: WEB-INF/web.xml(in = 1060) (out= 373)(deflated 64%)
WAR file generated.
```

```
The following WAR file was generated:
 C:\MFSXMLUtility\WAR\PB.war
Step 3 completed.
```

You can run step 2 several times to create multiple instance servlets and web.xml files. The WAR file that is created in step 3 contains all of the items that are generated during the multiple runs of step 2. For example:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Please enter your selection here: 3
Step 3: Generate WAR (Web Application aRchive) file containing one or more instance servlets
        In order to generate WAR file, running through step 2 in advance is mandatory.
        This step requires the following information:
        -Previously generated instance servlet class file(s) in the WEB-INF\classes directory
        -Previously generated depolyment descriptor (web.xml) in the WEB-INF directory

        *Please examine the content of the web.xml file in C:\MFSXMLUtility\WEB-INF\ and make
         any necessary additions.*
        *Note that the only web.xml file that will be packaged into the WAR file is in
         C:\MFSXMLUtility\WEB-INF\

This WAR file is going to be generated with the following instance servlets.
1) .\WEB-INF\classes\host1.class
2) .\WEB-INF\classes\host1.java
3) .\WEB-INF\classes\host2.class
4) .\WEB-INF\classes\host2.java
5) .\WEB-INF\classes\host3.class
6) .\WEB-INF\classes\host3.java

Enter the name of this WAR file: MYCOMPANY
Do you want to package additional files such as pictures with this WAR file? (y|n; default is no)
added manifest
adding: WEB-INF/ (in=0)(out=0)(stored 0%)
adding: WEB-INF/classes/ (in=0)(out=0)(stored 0%)
adding: WEB-INF/classes/host1.class (in=371)(out=262)(stored 29%)
adding: WEB-INF/classes/host1.java (in=609)(out=397)(stored 34%)
adding: WEB-INF/classes/host2.class (in=371)(out=263)(stored 29%)
adding: WEB-INF/classes/host2.java (in=614)(out=402)(stored 34%)
adding: WEB-INF/classes/host3.class (in=371)(out=263)(stored 29%)
adding: WEB-INF/classes/host3.java (in=609)(out=397)(stored 34%)
adding: WEB-INF/web.xml (in=3139)(out=473)(stored 84%)
WAR file generated.

The following WAR file was generated:
 C:\MFSXMLUtility\WAR\MYCOMPANY.war
Step 3 completed.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Figure 5. WAR file created with multiple servlets and web.xml files

Here is an example web.xml file that is created in "Step 3: Generating the Web Application Archive (WAR) file" on page 22:

```
<!DOCTYPE web-app (View Source for full doctype...)>
- <web-app>
- <servlet>
<servlet-name>host1</servlet-name>
<servlet-class>host1</servlet-class>
- <init-param>
<param-name>hostName</param-name>
<param-value>host1.your.company.com</param-value>
</init-param>
- <init-param>
<param-name>serverPlatform</param-name>
```

```
<param-value>1</param-value>
 </init-param>
- <init-param>
<param-name>dataStore</param-name>
<param-value>IMS1</param-value>
</init-param>
- <init-param>
<param-name>portNumber</param-name>
<param-value>335</param-value>
</init-param>
- <init-param>
<param-name>MFSXMIRepositoryURI</param-name>
<param-value>file:/c:\xmi</param-value>
</init-param>
- <init-param>
<param-name>MFSStyleSheet</param-name>
<param-value>file:/c:\$Projects\MFSXML\source\sample3270.xsl</param-value>
</init-param>
</servlet>
- <init-param>
<param-name>traceLevel</param-name>
<param-value>3</param-value>
</init-param>
-<init-param>
<param-name>executionTimeout</param-name>
<param-value>5000</param-value>
-</init-param>
- <init-param>
<param-name>socketTimeout</param-name>
<param-value>3000</param-value>
- </init-param>
- <servlet>
<servlet-name>host2</servlet-name>
<servlet-class>host2</servlet-class>
- <init-param>
<param-name>hostName</param-name>
<param-value>host2.your.company.com</param-value>
</init-param>
- <init-param>
<param-name>serverPlatform</param-name>
<param-value>1</param-value>
</init-param>
- <init-param>
<param-name>dataStore</param-name>
<param-value>IMS1</param-value>
</init-param>
- <init-param>
<param-name>portNumber</param-name>
<param-value>335</param-value>
</init-param>
- <init-param>
<param-name>MFSXMIRepositoryURI</param-name>
<param-value>file:/c:\xmi</param-value>
</init-param>
- <init-param>
<param-name>MFSStyleSheet</param-name>
<param-value>file:/c:\$Projects\MFSXML\source\sample3270.xsl</param-value>
</init-param>
- <init-param>
<param-name>traceLevel</param-name>
<param-value>2</param-value>
</init-param>
- <init-param>
<param-name>executionTimeout</param-name>
<param-value>8000</param-value>
- </init-param>
<init-param>
```

```
<param-name>socketTimeout</param-name>
<param-value>0</param-value>
- </init-param>
- <init-param>
<param-name>userName</param-name>
<param-value>KEVIN</param-value>
</init-param>
- <init-param>
<param-name>password</param-name>
<param-value>SJDFL</param-value>
</init-param>
- <init-param>
<param-name>groupName</param-name>
<param-value>GROUP1</param-value>
</init-param>
</servlet>
- <servlet>
<servlet-name>host3</servlet-name>
<servlet-class>host3</servlet-class>
- <init-param>
<param-name>hostName</param-name>
<param-value>host3.your.company.com</param-value>
</init-param>
- <init-param>
<param-name>serverPlatform</param-name>
<param-value>1</param-value>
</init-param>
- <init-param>
<param-name>dataStore</param-name>
<param-value>IMS1</param-value>
</init-param>
- <init-param>
<param-name>portNumber</param-name>
<param-value>335</param-value>
</init-param>
- <init-param>
<param-name>MFSXMIRepositoryURI</param-name>
<param-value>file:/c:\xmi</param-value>
</init-param>
- <init-param>
<param-name>MFSStyleSheet</param-name>
<param-value>file:/c:\$Projects\MFSXML\source\sample3270.xsl</param-value>
</init-param>
- <init-param>
<param-name>traceLevel</param-name>
<param-value>1</param-value>
</init-param>
</servlet>
- <init-param>
<param-name>executionTimeout</param-name>
<param-value>10000</param-value>
- </init-param>
- <init-param>
<param-name>socketTimeout</param-name>
<param-value>1500</param-value>
- </init-param>
- <servlet-mapping>
<servlet-name>host1</servlet-name>
<url-pattern>/host1</url-pattern>
</servlet-mapping>
- <servlet-mapping>
<servlet-name>host2</servlet-name>
<url-pattern>/host2</url-pattern>
</servlet-mapping>
- <servlet-mapping>
```

```
<servlet-name>host3</servlet-name>
<url-pattern>/host3</url-pattern>
</servlet-mapping>
</web-app>
```

(Optional) Step 4: Uploading WAR and XMI files using an FTP client

The MFS XML Utility provides an FTP client for uploading Web Application Archive (WAR) and XML Metadata Interchange (XMI) files onto the WebSphere Application Server. To use the FTP client you must know the hostname, user ID, password, destination server path name, and directory that contains files or file locations to upload to.

To upload WAR and XMI files:

1. From the MFS XML Utility window, choose selection 4 and press **Enter**:

   ```
   Choose services available to execute from below.
   1) Generate XMI files from MFS source files
   2) Generate instance servlet and WAS deployment descriptor
   3) Generate J2EE compliant WAR (Web application ARchive) file
   4) Uploading WAR and XMI files using FTP client
   5) Run previously saved batch file
   6) Exit

   Please enter your selection here: 4
   ```

2. If you meet the prerequisites, indicate if you want to continue by entering `y` (yes) or `n` (no) and then press **Enter**:

   ```
   Welcome to the FTP client to MFS XML Utility!
   Prior to using FTP client, note the following:
           1) Ensure that the necessary path to the FTP files exists. Create the
              path if necessary.
           2) Ensure that the correct permission settings for the directories
              and files are in place. Consult System administrator or system
              programmer should you have any questions.

   Continue (y|n; default is yes)? y
   ```

3. Specify the hostname and press **Enter**:

   ```
   Enter hostname: testing123
   ```

4. Specify your user ID and press **Enter**:

   ```
   Enter user ID: johndoe
   ```

5. Enter your password and press **Enter**:

   ```
   Enter password:
   ```

6. Specify the destination server path where you want to store the XMI files, or leave this entry blank to accept the default, and press **Enter**:

   ```
   Enter destination server path to place the xmi file(s) or press ENTER for default server
   directory:
   ```

   Note: The preexisting server path can be the same as the XMI repository URI specified in step 2 on WebSphere Application Server. You might need to configure the FTP server for this to work. For example, the preexisting server path or default server path could be the XMI repository URI like: `c:\xmi`.

7. Specify the XMI files (separated by space) or directory to upload and then press **Enter**:

   ```
   Specify xmi directory or xmi file(s) to upload: test.xmi
   You selected single XMI file for uploading.
   uploading test.xmi
   ```

(Optional) Step 5: Running a batch file

The MFS XML Utility provides the option to use a previously-saved batch file for "Step 1: Generating XMI files from MFS source files" on page 16 and "Step 2: Generating the instance servlet and the web.xml files" on page 20, instead of repeatedly invoking identical step 1 and step 2 runs.

To use a previously-saved batch file:

1. From the MFS XML Utility window, choose selection 5 and press **Enter**:

```
Choose services available to execute from below.
1) Generate XMI files from MFS source files
2) Generate instance servlet and WAS deployment descriptor
3) Generate J2EE compliant WAR (Web application ARchive) file
4) Uploading WAR and XMI files using FTP client
5) Run previously saved batch file
6) Exit

Please enter your selection here: 5
```

2. Specify the batch files to use, separated by a space, and press **Enter**:

```
Enter batch files separated by space: .\PHONEBOOK\PhoneBook.txt
RUNNING BATCH FILE .\PHONEBOOK\PhoneBook.txt
```

3. The MFS XML Utility goes through steps 1 and 2, using the values that are supplied in the batch file:

```
Beginning Step 1: Generating XMI files
Starting Step 2: Generate and compile instance servlet(s)
Generating servlet complete
Servlet is being compiled...
Compilation completed. Generating PhoneBookServlet.class

Moving generated instance servlet class file to WEB-INF directory...
Starts to put files in the WEB-INF directory...
Servlet is being compiled...
Compilation completed. Generating PhoneBookServlet.class.

Generating servlet deployment descriptor....
Instance servlet deployment descriptor files created.
```

For more information, see "Invoking the MFS XML Utility in batch mode" on page 10.

Chapter 4. MFS XML Utility logging

Device Types logging

Device type logs will be created after each step 1 run in `$MFSXMLUTILITY_HOME\logs\device_types_logs`. `$MFSXMLUTILITY_HOME` is the directory where MFS XML Utility is installed.

It will record all the device types contained in each of the MFS source files that are parsed. The format of the device types logging will be as follows:

```
For $EXAMPLE.mfs, the following device types are found:
1) $TYPE1
2) $TYPE2
...
```

The following is an example device types log from a run through step 1 that parses many MFS source files at once:

```
For C:\dfsivf2.mfs, the following device types are found:
1) 3270,2

For C:\dfsivf2m.mfs, the following device types are found:
1) 3270,2

For C:\dfsivf2_nextpp.mfs, the following device types are found:
1) 3270,2

For C:\dfsivf2_unsupported_PF.mfs, the following device types are found:
1) 3270,2

For C:\dfsivf34.mfs, the following device types are found:
1) 3270,2

For C:\IGL.EDVR.PROD060.MFS, the following device types are found:
1) 3270,2
2) 3270-A2

For C:\IGL.EDVR.PROD101.MFS, the following device types are found:
1) 3270,2
2) 3270-A2

For C:\IGL.EDVR.PROD105.MFS, the following device types are found:
1) 3270,2
2) 3270-A2

For C:\MFSO5E.mfs, the following device types are found:
1) 3270,2

For C:\MFSO5EPFK.mfs, the following device types are found:
1) 3270,2

For C:\MFSPF.mfs, the following device types are found:
1) 3270,2

For C:\oe4cnio1.mfs, the following device types are found:
1) 3270,2

For C:\oe4cnio1x.mfs, the following device types are found:

For C:\OE4COR01.mfs, the following device types are found:
1) 3270-A2
2) 3270P,2
```

```
For C:\pbliteral.mfs, the following device types are found:
1) 3270-A02

For C:\pwrd2.mfs, the following device types are found:
1) 3270,2

For C:\TTT3D.MFS, the following device types are found:
1) 3270,2
```

MFS Importer Parse logging

The MFS Importer parse logs (in Step 1) will be created for each run of step 1 in `$MFSXMLUTILITY_HOME\logs\parse_logs`.

This parse log file will contain detailed information if the parsing of any specific MFS source files resulted in warnings or errors. If the parsing resulted in warnings, the XMI files can still be generated without errors. However, if an MFS source file is parsed with errors, then the XMI files will not be generated. Refer to *MFS XML Utility User's Guide* for a more detailed explanation of the "IXFI" prefixed messages. A common message will look like `IXFI008W: An unsupported device type was found: $UNKNOWN_DEVICE`.

Here is an example of the parse.log:

```
Parse failed
C:\dfsivf2.mfs parsed successfully
C:\dfsivf2m.mfs parsed successfully
C:\dfsivf2_nextpp.mfs parsed successfully
C:\dfsivf2_unsupported_PF.mfs parsed successfully
C:\dfsivf34.mfs parsed successfully
C:\dfsivf34M.mfs Returned a parse error: com.ibm.etools.mfs.importer.TokenMgrError: Lexical error
at line 80, column 69.  Encountered: "P" (80), after : ""C:\IGL.EDVR.PROD060.MFS Returned a
warning message: IXFI008W: An unsupported device type was found: SCS1

IXFI008W: An unsupported device type was found: SCS1

C:\IGL.EDVR.PROD101.MFS Returned a warning message: IXFI008W: An unsupported device type was found: SCS1

IXFI008W: An unsupported device type was found: SCS1

C:\IGL.EDVR.PROD105.MFS Returned a warning message: IXFI008W: An unsupported device type was found: SCS1

IXFI008W: An unsupported device type was found: SCS1

C:\MFSO5E.mfs parsed successfully
C:\MFSO5EPFK.mfs parsed successfully
C:\MFSPF.mfs parsed successfully
C:\oe4cnio1.mfs parsed successfully
C:\oe4cnio1x.mfs Returned a warning message: IXFI008W: An unsupported device type was found: 2740,2

IXFI008W: An unsupported device type was found: 2740,2

C:\OE4COR01.mfs parsed successfully
C:\pbliteral.mfs parsed successfully
C:\pwrd2.mfs parsed successfully
C:\TTT3D.MFS parsed successfully
```

Chapter 5. MFS XML Utility Messages and Codes

This section describes the messages issued by the MFS XML Utility.

When contacting the IBM Support Center, please include the log directory from the MFS XML Utility installation directory.

IXFU001I

`Batch file <name> cannot be found`

Explanation

The batch files that you specified cannot be found.

System action

Returns back to the main selection menu.

User response

Verify the location of the batch file and try again.

IXFU002E

`Error while initializing output path.`

Explanation

An error occurred when the output path was initialized. The standard output path should be the location of the MFS XML Utility installation directory (for example, `c:\MFSXMLUtility`).

System action

None.

User response

Check the current directory for permission setting issues to make sure that the directories are writable.

ICFU003I

`MFS Source file not found.`

Explanation

The MFS source files that you specified cannot be found.

System action

The MFS XML Utility displays the message `parse failed` and returns to the main selection menu.

User response

Verify whether the MFS source file is in the path that is recorded in the parse.log file. Verify the location of the MFS source file and try again.

IXFU004E

```
Parse exception. See parse.log
```

Explanation

Exceptions that are thrown by the MFS Importer during parsing are generally a form of a non-fatal warning message. For example, parse exceptions include errors loading external references, unsupported device types, and unresolved relationships.

System action

The MFS XML Utility displays the exception thrown by the MFS Importer.

User response

Verify that the MFS source file contains the correct information and the external references can be resolved.

IXFU005E

```
Throwable Parse error. See parse.log.
```

Explanation

Errors that are thrown by the MFS Importer usually are related to the correctness of the syntax of the MFS source files.

System action

The MFS XML Utility displays the error.

User response

Verify and correct the MFS source files. See the MFS chapter in *IMS Application Programming: Transaction Manager* if you require additional information.

IXFU006E

```
Non-throwable parse error. See parse.log.
```

Explanation

This is the error message for all of the MFS Importer errors that are not parse exception or parse errors.

System action

The MFS XML Utility displays the error.

User response

Verify and correct the MFS source files. See the MFS chapter in the *IMS Application Programming: Transaction Manager* if you require additional information.

IXFU007E

```
Cannot find the parse.log file after creation
```

Explanation

The parse.log file cannot be found in the default directory.

System action

The MFS XML Utility displays an error message.

User response

The parse.log file is created after an MFS source file is parsed. Check the MFS XML Utility installation directory to see if the parse.log file was removed or deleted. If the file was deleted by mistake, you can recreate a blank parse.log file and store it in the MFS XML Utility installation directory.

IXFU008E

```
Unable to create or write to parse.log
```

Explanation

An error occurred during the parse.log file generation or an error occurred when the log was being written in to the parse.log file.

System action

The MFS XML Utility displays an error message.

User response

Verify the permission setting of the directory in which the MFS XML Utility runs. Make sure the file is not READ-ONLY. If the problem persists, exit the utility, rename the current parse.log file, recreate a blank parse.log file, and re-run the utility.

IXFU009E

```
Exception thrown while retrieving device feature
```

Explanation

A possible error with the MFS source file occurred.

System action

The system exits immediately.

User response

Verify and make sure that the MFS source files are valid. See the MFS chapter in *IMS Application Programming: Transaction Manager* if you require additional information.

IXFU010E

`Exception thrown from getFeatures() while parsing the MFS source files`

Explanation

This error is related to the device feature section in the MFS source files.

System action

The system exits immediately.

User response

Verify and correct the device features section of the MFS source files.

IXFU011E

`Invalid device type selection`

Explanation

An IO exception was thrown while reading the device type selection cannot be retrieved.

System action

The MFS XML Utility message displays.

User response

Check and verify the correctness of the format of the device type selection.

IXFU012E

`Invalid device feature selection`

Explanation

The device feature selection cannot be retrieved.

System action

The MFS XML Utility message displays.

User response

Check and verify the correct format of the device feature selection.

IXFU013E

`IO error happened when reading the MFS source files user specified`

Explanation

An IO exception was thrown while reading the input to the MFS source file.

System action

The MFS XML Utility displays an error message.

User response

Check and verify that no other users are updating or deleting the files while the utility is processing. Re-run the step.

IXFU014I

```
IO error trying to read Device Characteristics Table
```

Explanation

An IO exception while reading the optional DCT (Device Characteristics Table).

System action

The MFS XML Utility displays an error message.

User response

Check the location and verify the correctness of the DCT. The file format should be in binary. You can also regenerate the DCT using the IMS MFS Reversal Utility.

IXFU015I

```
Invalid binary mode selection
```

Explanation

A value other than " Y" or "N" was specified. The default is no.

System action

The MFS XML Utility selects non-binary mode (the default value) and continues.

User response

Re-run "Step 1: Generating XMI files from MFS source files" on page 16 and select the correct binary mode.

IXFU016E

```
Invalid source codepage
```

Explanation

The source codepage specified cannot be found. See "Host codepage" on page 47 for a list of supported codepages.

System action

The MFS XML Utility displays an error message.

User response

Check and verify that the correct codepage is used.

IXFU017E

`Invalid host codepage`

Explanation

The host codepage specified cannot be found. See “Source codepage” on page 48 for a list of supported codepages.

System action

The MFS XML Utility displays an error message.

User response

Check and verify that the correct codepage is used.

IXFU018E

`Null pointer exception occurred while serializing into EMF XMI files.`

Explanation

This error is associated with the MFSParser.java file within the MFS Importer.

System action

The system exits immediately.

User response

Retry again and contact the IBM Support Center.

IXFU019E

`IO exception occurred while serializing into EMF XMI files.`

Explanation

An IO exception occurred while generating the XMI files.

System action

The system exits immediately.

User response

Check and verify the permission to be writable issues of the XMI output directory.

IXFU020E

`General exception (not nullpointerexception) occurred while serializaing into EMF XMI files.`

Explanation

An exception, other than nullpointer and IO, occurred during serialization.

System action

The system exits immediately.

User response

Contact the IBM Support Center.

IXFU021E

`IO exception occurred while retrieving current directory information.`

Explanation

This error occurs when the MFS XML Utility tries to get information about its current directory.

System action

An error message is displayed.

User response

Check for the directory permission-related setting and make sure the directory is readable.

IXFU022E

`IO Exception occurred during generation of web.xml.`

Explanation

An IO error occurred during the generation of the web.xml file.

System action

An error message is displayed.

User response

The directory permission-related setting must allow updates. If the setting is set correctly and the error persists, contact the IBM Support Center.

IXFU023E

`IO Exception occurred during generation of web.xml.`

Explanation

An IO error occurred during the generation of the AllWeb.xml file in the installation directory in which the MFS XML Utility runs.

System action

The MFS XML Utility displays an error message.

User response

Check for directory permission-related setting. If the setting is set correctly and the error persists, contact the IBM Support Center.

IXFU024I

```
Error moving generated xmi file to user specified directory.
```

Explanation

The specified output directory does not exist.

System action

XMI files are generated to the MFS XML Utility installation directory.

User response

Ensure that the directory you specified exists. If not, create the directory structure and re-run the step. Contact the IBM Service Center if the problem persists.

IXFU025E and IXFU026E

```
IO error trying to read servlet arguments in expert mode. This error should not occur.
```

Explanation

This is an IO error.

System action

The MFS XML Utility displays an error message.

User response

Try again if the issues persists contact the IBM Support Center.

IXFU027E

```
The servlet file cannot be found.
```

Explanation

The specified instance servlet file path does not exist.

System action

The MFS XML Utility displays an error message.

User response

Verify that all of the path information is correct and make sure that another user is not renaming or deleting the MFS XML Utility directory at the same time.

IXFU028E

```
Unable to write to the servlet file.
```

Explanation

IO exception during creating or writing the servlet file.

System action

The MFS XML Utility displays an error message.

User response

Correct the permission to be not read only.

IXFU029E

```
Exception occurred during servlet compilation. System error.
```

Explanation

The process executing the Java compiler did not return.

System action

The system deletes the generated instance servlet Java file.

User response

Rerun "Step 2: Generating the instance servlet and the web.xml files" on page 20. You might also need to restart JVM. Verify that JVM is at least JDK version 1.4 or higher.

IXFU030E

```
Runtime execution of javac error. Some other processes might be changing the environment
variable at the same time.
```

Explanation

This is a Runtime.exec() error, possibly because it cannot find javac.

System action

The MFS XML Utility displays an error message.

User response

Re-run “Step 2: Generating the instance servlet and the web.xml files” on page 20. You might also need to restart the and re-run the MFS XML Utility.

IXFU031E

`The AllWeb.xml in the installation directory of MFS XML Utility is missing.`

Explanation

The `AllWeb.xml` file in the directory where the MFS XML Utility runs does not exist and might have been deleted.

System action

The MFS XML Utility displays an error message.

User response

Replace the AllWeb.xml file or create your own AllWeb.xml file by merging all of the web.xml files in each output directory. Re-run “Step 2: Generating the instance servlet and the web.xml files” on page 20 afterward.

IXFU032E

`Error saving user defaults to the user.default file in the installation directory.`

Explanation

An IO exception occurred from the user.default file in the MFS XML Utility directory.

System action

The user defaults are not saved.

User response

Verify that the user.default file exists.

IXFU033E

`error reading user defaults from the user.default file.`

Explanation

An IO exception occurred from the user.default file in the MFS XML Utility directory.

System action

The user defaults were not loaded.

User response

Verify that the user.default file exists and is readable.

IXFU034I

`There is no instance servlet class files ready for packaging into WAR file.`

Explanation

Both instance servlet class files and the web.xml file with the deployment information are required.

System action

The MFS XML Utility returns back to the selection menu.

User response

Generate instance servlets first by running step 2 or by manually adding class files to `$INSTALLATION_DIRECTORY\WEB-INF\classes` and update the web.xml file in `$INSTALLATION_DIRECTORY\WEB-INF`.

IXFU035I

`WEB-INF directory missing`

Explanation

You might have deleted the WEB-INF directory that is generated by the MFS XML Utility from the installation directory in which the utility runs.

System action

The MFS XML Utility displays instructions that explain how to construct the WEB-INF directory.

User response

Create the directory first and re-run "Step 2: Generating the instance servlet and the web.xml files" on page 20 to regenerate the instance servlets and associated web.xml file.

IXFU036E

`Process interrupted error occurred during runtime execution of JAR command inside Java 2 SDK.`

Explanation

A process-interrupted error occurred during the runtime execution of a JAR command inside the Java 2 SDK.

System action

The MFS XML Utility displays an error message.

User response

Exit and restart the MFS XML Utility first and then re-run "Step 3: Generating the Web Application Archive (WAR) file" on page 22 and make sure that no other interfering process is running at the same time.

IXFU037E

`Error occurred during runtime execution of JAR command inside Java 2 SDK.`

Explanation

This is a runtime.exec() error.

System action

The MFS XML Utility displays an error message.

User response

Exit and restart the MFS XML Utility first and then re-run "Step 3: Generating the Web Application Archive (WAR) file" on page 22.

IXFI001E

`An MFS file [THE_COPY_SOURCE] required by a COPY statement was not found in the specified MFS directory [THE_MFS_SOURCE_DIRECTORY],`where *THE_COPY_SOURCE* and *THE_MFS_SOURCE_DIRECTORY* are both variables.

Explanation

An MFS file required by a COPY statement was not found in the specified directory.

System action

The message is issued, and the parser is stopped.

User response

Copy the missing MFS file into the specified directory.

IXFU002E

`Error while initializing output path.`

Explanation

An error occurred when the output path was initialized. The standard output path should be the location of the MFS XML Utility installation directory (for example, `c:\MFSXMLUtility`).

System action

None.

User response

Check the current directory for permission setting issues to make sure that the directories are writable.

IXFI003W

`The device characteristics file could not be opened.`

Explanation

The importer could not open the device characteristics file for reading.

System action

The message is issued, and execution continues.

User response

Make sure that the device characteristic file exists and has the correct file access mode.

IXFI004W

`The device characteristics file was invalid.`

Explanation

An I/O exception occurred while the device characteristics file was read.

System action

The message is issued, and execution continues.

User response

Make sure that the contents of the device characteristic file are correct and in binary format.

IXFI005W

`The external URI is invalid.`

Explanation

A midname, modname or table name was expected while an external reference for the NXT or OCT parameter was loading. The midname, modname, or table name is missing.

System action

The message is issued, and Importer generates an empty default reference for the midname, modname, or table name to resolve the relationship. Execution continues.

User response

A forward reference in the MFS source files can produce this message. This error occurs when information needed to complete the parsing of the MFS source resides in another file. Ensure that the associated xmi file for the MID, MOD or TABLE name in the warning is produced and fully populated (not empty).

IXFI006W

`A default object was generated.`

Explanation

This error refers to an unresolved relationship. This error occurs when the parser needs to generate an empty XMI file so that cross XMI file relationships can be set correctly.

System action

The message is issued, and execution continues.

User response

Correct the error in the MFS source and restart the importer.

IXFI007W

```
An unresolved relationship occurred.
```

Explanation

This error refers to an unresolved relationship. This error occurs when the information provided in the source file is incorrect. For example, when an MFLD references a non-existent or invalid DFLD.

System action

The message is issued, and execution continues.

User response

Correct the error in the MFS source and restart the importer.

IXFI008W

```
The device is unsupported.
```

Explanation

A non-3270 device type was found in the source.

System action

The message is issued, and processing continues.

User response

None.

IXFI009E

```
The encoding is unsupported.
```

Explanation

An unsupported encoding was specified.

System action

The message is issued, and the importer stops.

User response

Choose a supported encoding from the drop-down list.

IXFI010W

`The parser overwrote an XMI file.`

Explanation

The MFS source contained a definition for a MID/MOD or TABLE statement that was already defined in the XMI repository. The new definition will overwrite the old definition.

System action

The message is issued, and processing continues.

User response

None.

Chapter 6. Codepages

This topic lists the available source and host codepages.

The following topics provide additional information:

- "Host codepage"
- "Source codepage" on page 48

Host codepage

The host codepage is the EBCDIC codepage that the MFS source was encoded in. The following list shows the codepages that are available.

Host codepage	Description
Cp037	EBCDIC United States
Cp273	EBCDIC Germany
Cp277	EBCDIC Denmark, Norway
Cp278	EBCDIC Finland, Sweden
Cp280	EBCDIC Italy
Cp284	EBCDIC Spain, Latin America
Cp285	EBCDIC UK, Ireland
Cp297	EBCDIC France
Cp420	EBCDIC Arabic
Cp424	EBCDIC Hebrew
Cp500	EBCDIC Latin 1
Cp838	EBCDIC Thai
Cp870	EBCDIC Latin 2
Cp871	EBCDIC Iceland
Cp875	EBCDIC Greek
Cp918	EBCDIC Urdu
Cp924	EBCDIC Latin 9
Cp930	EBCDIC Japan DBCS
Cp933	EBCDIC Korea DBCS
Cp935	EBCDIC China DBCS
Cp937	EBCDIC Taiwan DBCS
Cp939	EBCDIC Japan Extended DBCS
Cp1025	EBCDIC Cyrillic
Cp1026	EBCDIC Latin 5 (Turkey)
Cp1046	EBCDIC Arabic 8 bit
Cp1047	EBCDIC Open Edition
Cp1097	EBCDIC Farsi
Cp1112	EBCDIC Baltic

Host codepage	Description
Cp1122	EBCDIC Estonia
Cp1123	EBCDIC Ukraine
Cp1140	ECECP United States
Cp1141	ECECP Germany
Cp1142	ECECP Denmark, Norway
Cp1143	ECECP Finland, Sweden
Cp1144	ECECP Italy
Cp1145	ECECP Spain
Cp1146	ECECP UK, Ireland
Cp1147	ECECP France
Cp1148	ECECP Multilingual
Cp1149	ECECP Iceland
Cp1364	EBCDIC Korea KS extended
Cp1371	EBCDIC Taiwan with Euro
Cp1388	EBCDIC China GBK
Cp1390	EBCDIC Japan Katakana Euro
Cp1399	EBCDIC Japan Latin with Euro

Source codepage

In addition to the source codepage, you also need to identify the download format in which the sources were downloaded from the host. An MFS source file can be downloaded either in binary or text format. However, if a MFS source contains DBCS characters, then the source must be downloaded in binary format. Downloading a DBCS file in text format will result in a parser error at later stage when the source is parsed. For the source files downloaded in text format, you need to select an additional workstation text codepage. The following list shows the available single-byte text codepages for non-DBCS source parsing:

Source codepage	Description
ASCII	ASCII 7 bit
Cp437	PC United States
Cp737	MS-DOS Greek
Cp775	MS-DOS Baltic Rim
Cp850	PC Latin 1
Cp852	PC Latin 2
Cp855	PC Cyrillic
Cp856	PC Hebrew (old)
Cp857	PC Latin 5
Cp858	PC Latin 1 (euro)
Cp859	PC Latin 9
Cp860	PC Portugal
Cp861	PC Iceland
Cp862	PC Israel

Source codepage	Description
Cp863	PC Canadian French
Cp864	PC Arabic
Cp865	PC Nordic
Cp866	PC Russia
Cp867	PC Israel
Cp868	PC Urdu
Cp869	PC Greece
Cp874	PC Thai
Cp921	PC Baltic
Cp922	PC Estonian
Cp923	PC Latin 9
Cp964	EUC Taiwan
Cp970	EUC Korea
Cp1006	Urdu 8 bit
Cp1098	PC Farsi
Cp1124	PC Ukraine
Cp1250	Windows Latin 2
Cp1251	Windows Cyrillic
Cp1252	Windows Latin 1
Cp1253	Windows Greek
Cp1254	Windows Latin 5 (Turkey)
Cp1255	Windows Hebrew
Cp1256	Windows Arabic
Cp1257	Windows Latin 4 (Baltic)
Cp1258	Windows Vietnamese
Cp1383	EUC China
Cp33722	EUC Japan
Cp33722C	EUC Japan syntax
EUC_CN	EUC China
EUC_JP	EUC Japan
EUC_KR	EUC Korea
EUC_TW	EUC Taiwan
ISCII91	ISCII Hindi 91
ISO8859_1	ISO Latin 1
ISO8859_15_FDIS	ISO Latin 9
ISO8859_2	ISO Latin 2
ISO8859_3	ISO Latin 3
ISO8859_4	ISO Latin 4
ISO8859_5	ISO Cyrillic
ISO8859_6	ISO Arabic
ISO8859_7	ISO Greek

Source codepage	Description
ISO8859_8	ISO Hebrew
ISO8859_9	ISO Latin 5
JIS0201	Japan JIS 0201
JIS0208	Japan JIS 0208
JIS0212	Japan JIS 0212
Johab	PC Korean
KOI8_R	Russia Internet
TIS620	Thailand

Chapter 7. Notices

This information was developed for products and services offered in the U.S.A.

IBM® may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM® representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental. COPYRIGHT LICENSE: This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

(your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. Copyright IBM Corp. 2003, 2005. All rights reserved. Trademarks

Chapter 8. Trademarks

IBM, IMS, WebSphere, and z/OS are trademarks of International Business Machines Corporation in the United States, other countries, or both.

Microsoft® and Windows® are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.