

IBM[®] WebSphere[®] Commerce



Catalog Manager User's Guide

Version 54

IBM[®] WebSphere[®] Commerce



Catalog Manager User's Guide

Version 54

Note

Before using this information and the product that it supports, read the information in “Notices” on page 113.

First Edition, Third Revision (September 2002)

This edition applies to the following product:

- IBM WebSphere Commerce, Version 5.4 (Program 5724-A18)

and to all subsequent releases and modifications of the above listed products until otherwise indicated in new editions. Make sure that you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office that serves your locality. Publications are not stocked at the address given below.

IBM welcomes your comments. You can send your comments by any one of the following methods:

1. Electronically to the e-mail address listed below.

torrcf@ca.ibm.com

Be sure to include your entire network address if you wish a reply.

2. By mail to the following address:

IBM Canada Ltd. Laboratory
B3/KB7/8200/MKM
8200 Warden Avenue
Markham, Ontario, Canada L6G 1C7

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way that it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 2002. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Before you begin	v
Conventions used in this book	v
Who should read this book	vi
Where to find more information	vi

Part 1. Overview of the Catalog Manager 1

Part 2. Transforming, loading, and extracting data 3

Chapter 1. Introduction	5
Utilities	5
Administrative tools	8

Chapter 2. Transforming text 9

Launching the Text Transformation tool	10
Using the Text Schema Edit View	11
Creating a schema file	11
Opening a schema file	11
Saving a schema file	11
Editing a schema file	11
Adding an element	11
Removing an element	12
Replacing an element	12
Moving an element one row up	12
Moving an element one row down	12
Adding an attribute	12
Removing an attribute	12
Replacing an attribute	12
Moving an attribute one row up	12
Moving an attribute one row down	12
Changing a schema file structure	13
Preparing a schema file to transform data from an XML to a character-delimited variable format	13
Using the Transformation Command Edit View	14
Creating a command	14
Removing a command	14
Editing or replacing a command	14
Moving a command one row up	14
Moving a command one row down	15
Clearing a command	15
Using the Transformation Process View	15

Chapter 3. Transforming XML data 17

Launching the XSL Editor	18
Working with mapping rule building areas	19
Creating a mapping rule building area	19
Modifying a mapping rule building area	19
Deleting a mapping rule building area	19
Using the XSL Editor	20
Creating a mapping rule	20
Element-to-element mapping	20

Attribute-to-attribute mapping	20
Creating a custom mapping expression	20
Deleting a mapping rule	21
Processing an XML transformation	21
Customizing the Mapping Rule table	21
Displaying a complete XSL rule/value expression	21

Chapter 4. Generating a DTD and schema 23

Setting up the DTD Generator	23
Generating a DTD	24
Generating a schema and a detailed XML file	26

Chapter 5. Resolving identifiers 27

Setting up the ID Resolver	27
Setting how the ID Resolver handles timestamps	28
Setting how the ID Resolver handles storage	28
Setting how the ID Resolver handles database drivers	29
Determining how to process data	29
Choosing the load method	29
Choosing the update method	30
Choosing the mixed method	30
Using the ID-resolution techniques	31
Specifying a properties file with the ID Resolver	31
Using a properties file to generate identifiers	32
Using a properties file with compound keys	33
Using a properties file with cascaded primary keys	34
Using internal-alias resolution	35
Partial example of using internal-alias ID resolution	36
Using unique-index resolution	36
Partial example of unique-index resolution	37
Loading data into the MEMBER table	38
Creating a foreign relationship using the REFKEYS table	39
Troubleshooting errors	40

Chapter 6. Loading data 41

Setting up the Loader	41
Ignoring elements in the input file	42
Inserting NULL into a column	42
Loading timestamps and date data	42
Loading current timestamps	43
Example of loading current timestamps	45
Examples of adding durations to current timestamps	45
Managing event queues	46
Running with different database software and operating systems	46
Substituting a component	49
Using Product Advisor search-space synchronization	50

Customizing Product Advisor search-space synchronization	53
Determining how to process data when using the Loader	54
Choosing the load method	54
Choosing the import method	55
Choosing the SQL import method	55
Other considerations	55
Loading large documents.	56
Troubleshooting tip.	57

Chapter 7. Extracting data 59

Creating an extraction filter	59
Setting up the Extractor	62

Chapter 8. Using the Loader package logger 63

Configuring logging in your environment for Windows NT, Windows 2000, AIX, Linux, and Solaris systems	63
Example of setting the classpath variable	63
Example of specifying the com.ibm.wca.logging.configFile system property	63
Customizing logging for the Loader package	64
Handlers	64
Filters	66
Formats	66
Example: WCALoggerConfig.xml and WCALogger.dtd	67
WCALoggerConfig.xml	67
WCALogger.dtd	68

Chapter 9. Using the Loader package error reporter 69

Chapter 10. Configuring Loader package commands and scripts. 71

Part 3. Using the Web editor 73

Chapter 11. Setting up the Web editor 75

Configuring the Web editor	76
Editing the webeditor.properties file	76
Changing the location of the temporary files	77
Creating an XML form-description file using the DTD Generator	78
Customizing the XML form description	80
Editing form names	82

Changing a field description.	83
Adding a drop-down menu	83
Adding field help	84
Customizing search results and the work-session list	85
Editing the weProcessList file	85
Editing the webeditor.xml file	88

Chapter 12. Working with catalogs. 89

Adding a record to a table using the Web editor	89
Modifying a record in a table using the Web editor	90
Deleting a record from a table using the Web editor	91

Part 4. Command Reference 93

Chapter 13. DTD Generate command 95

DTD Generate command for Windows, AIX, Linux, and Solaris systems.	95
DTD Generate command for iSeries systems	97

Chapter 14. Extract command. 99

Extract command for Windows, AIX, Linux, and Solaris systems	99
Extract command for iSeries systems	100

Chapter 15. ID Resolve command. 101

ID Resolve command for Windows, AIX, Linux, and Solaris systems	101
ID Resolve command for iSeries systems	103

Chapter 16. Load command 105

Load command for Windows, AIX, Linux, and Solaris systems	105
Load command for iSeries systems	107

Chapter 17. Text Transform command 109

Text Transform command for Windows, AIX, Linux, and Solaris systems	109
Text Transform command for iSeries systems	110

Chapter 18. XML Transform command 111

XML Transform command for Windows, AIX, Linux, and Solaris systems	111
XML Transform command for iSeries systems	112

Notices 113

Trademarks and service marks.	115
---------------------------------------	-----

Before you begin

The *IBM WebSphere Commerce Catalog Manager User's Guide* provides information about the WebSphere Commerce Catalog Manager. In particular, it provides details on the following topics:

- Transforming, loading, and extracting data using the Catalog Manager's tools and utilities
- Using the Catalog Manager Web editor to work with catalog data
- Catalog Manager commands


Conventions used in this book


This book uses the following conventions:


Boldface type indicates commands or graphical user interface (GUI) controls such as names of fields, buttons, or menu choices.


Monospaced type indicates examples of text that you enter exactly as shown as well as file names and directory paths.


Italic type is used for emphasis and for variables for which you substitute your own values.


 indicates information that is specific to WebSphere Commerce for Windows®.


 indicates information that is specific to WebSphere Commerce for Windows NT®.


 indicates information that is specific to WebSphere Commerce for Windows 2000®.


 indicates information that is specific to WebSphere Commerce for AIX®.


 indicates information that is specific to WebSphere Commerce for Solaris™ Operating Environment software.


 indicates information that is specific to WebSphere Commerce for Linux.

 indicates information that is specific to WebSphere Commerce for IBM @server™ iSeries™ (formerly called AS/400®)

 indicates information that is specific to DB2® Universal Database.

 indicates information that is specific to the Oracle® database.

 indicates information that is specific to WebSphere Commerce Business Edition.

 indicates information that is specific to WebSphere Commerce Professional Edition.

Who should read this book

This book should be read by site developers, administrators, and contributors who need to understand how to use the WebSphere Commerce Catalog Manager as well as by other business users who need to understand its functions.

In particular, this guide should be read by WebSphere Commerce Store Developers and Site Administrators who need to understand how to use the various functions of the WebSphere Commerce Catalog Manager.

WebSphere Commerce Store Developers and Site Administrators who are performing programmatic extensions should also have knowledge in the following areas:

- Database technology
- Enterprise JavaBeans™ component architecture
- Hypertext Markup Language (HTML)
- Java™
- JavaServer Pages™ technology
- VisualAge® for Java™, Enterprise Edition, Version 3.5 or later
- Extensible Markup Language (XML)

Category and Product Managers should read this guide in order to understand how to communicate catalog, product, and item requirements to Store Developers. Category and Product Managers should also understand how the Web editor can be used for updating the store's catalog.

Where to find more information

This user's guide is available in Adobe Portable Document Format (PDF) from the WebSphere Commerce Web site. Please check the WebSphere Commerce Technical Library Web site for the most recent versions of this document and other information related to WebSphere Commerce:

-  http://www.ibm.com/software/webservers/commerce/wc_be/lit-tech-general.html
-  http://www.ibm.com/software/webservers/commerce/wc_pe/lit-tech-general.html

Part 1. Overview of the Catalog Manager

WebSphere Commerce Catalog Manager provides a generic toolkit that provides various functions that can be strung together in the required sequence to solve your particular catalog-management problems. It is flexible enough to handle customizations that are made to the WebSphere Commerce schema.

Catalog Manager provides the ability to aggregate information from multiple sources into a consolidated WebSphere Commerce system and to remap all that diverse data into a standard catalog- and product-definition format using XML files as the standard means of managing information.

Catalog Manager provides the means for you to do the following:

- Import data from multiple input sources in the form of ASCII and XML files into WebSphere Commerce
- Transform data from ASCII to XML format and back again
- Remap data from one XML format to another
- Aggregate data from multiple input streams into one aggregated database
- Create/edit/delete data through a Web-browser interface

Catalog Manager includes the following:

- Catalog Manager Loader package

This package consists primarily of command utilities for preparing and loading data into a WebSphere Commerce database. You can use the Loader package to load large amounts of data and to update data in your WebSphere Commerce database.

The Loader package also allows you to do the following:

- Extract data from a database as an XML document
- Transform XML data into alternate XML formats
- Transform data between a character-delimited variable format and an XML data format

- Catalog Manager administrative tools

Catalog Manager also includes the following two tools with a user interface to assist in the administration of its functions:

- Text Transformation tool
- XSL Editor

- Catalog Manager Web editor

The Web editor enables you to create, delete, and make changes to your catalog data through a Web browser.

Part 2. Transforming, loading, and extracting data

Chapter 1. Introduction

The Catalog Manager Loader package consists primarily of utilities for preparing and loading data into a WebSphere Commerce database. You can use the Loader package to load large amounts of data and to update data in your WebSphere Commerce database. The Loader package also allows you to do the following:

- Extract data from a database as an XML document
- Transform XML data into alternate XML formats
- Transform data between a character-delimited variable format and an XML data format

The Catalog Manager also includes the following two tools with a user interface to assist in the administration of its functions:

- Text Transformation tool
- XSL Editor

Utilities

The WebSphere Commerce Loader package includes utilities for preparing and loading data into a WebSphere Commerce database. You can use these utilities to load large and small amounts of data as well as to update data in your WebSphere Commerce database.

The loading process consists of the steps necessary in order to move data into your WebSphere Commerce database:

1. Generating a DTD using the DTD Generator
2. Resolving identifiers in the input files using the ID Resolver
3. Loading the data using the Loader

The Loader package also includes utilities for extracting data from a database as an XML document and for transforming XML data into alternate XML formats.

The Loader package consists of the following utilities:

- **Text Transformer**

The Text Transformer transforms data between a character-delimited variable format and an XML data format.

See Chapter 2, “Transforming text” on page 9 for more information.

- **XML Transformer**

The XML Transformer changes, aggregates, and remaps the data in an XML document to alternate XML formats for use by other users or systems as needed.

See Chapter 3, “Transforming XML data” on page 17 for more information.

- **DTD Generator**

The DTD specifies structural elements and markup definitions that can be used within an XML data document. For example, a DTD can list elements to be used in a document and specify the attributes that each element can take.

The DTD Generator is a utility that creates a DTD for the Loader to use based on the database schema. This DTD describes the tables and columns into which the Loader imports data. The DTD Generator can also create a schema and detailed XML document that can be used with the Catalog Manager Web editor.

The DTD Generator generates a DTD based on the target database to which your data must conform. This DTD will be used throughout the load process. The DTD Generator needs to be run only once.

See Chapter 4, “Generating a DTD and schema” on page 23 for more information.

- **ID Resolver**

The ID Resolver is a utility that generates identifiers for XML elements that require them. If your XML content already supplies identifiers, you do not have to run the ID Resolver.

The ID Resolver updates a set of XML elements with their associated identifiers. This step is essential because Loader package XML files map directly to the target database schema. As such, they must have identifiers.

The ID Resolver includes an error reporter that generates an exception document if there is an error.

See Chapter 5, “Resolving identifiers” on page 27 for more information.

- **Loader**

The Loader uses valid and well-formed XML as input to load data into the database. Elements of the XML document map to table names in the database, and element attributes map to columns. The Loader is the most common means of loading data into a system.

The Loader allows column-level updates to a table. It also allows you to delete data from a database.

The following example shows an excerpt from well-formed and valid XML input to the Loader:

```
<ADDRBOOK
  ADDRBOOK_ID="11801"
  MEMBER_ID="100"
  DISPLAYNAME="Friends"
  DESCRIPTION="All my friends"
  TYPE="P"
/>
```

In the above example, ADDRBOOK is the table name and the columns to be updated are indicated by attributes of the ADDRBOOK element.

The Loader utility includes the following features:

- **Error reporter**

The Loader includes an error reporter that generates an exception document if there is an error.

- **Product Advisor search-space synchronization**

Product Advisor search spaces are created by extracting information from the WebSphere Commerce catalog and representing that information in a form suitable for searching. The search spaces and the catalog become unsynchronized when the catalog is updated. To avoid delay in synchronizing the search spaces with the catalog, you can enable the Product Advisor search-space synchronization feature of the Loader.

If Product Advisor search-space synchronization is enabled, the Loader will schedule appropriate search-space commands by adding the command information to the scheduler job table SCHCONFIG and scheduler job-instance table SCHSTATUS as indicated in the following table.

Table	Mode	Search-Space command	Condition
CATENTRY	Update	UpdateSearchSpaces	Update publish flag
	Delete	RemoveProductFromAllSearchSpaces	Delete product
CATENTDESC	Add	UpdateSearchSpaces	Add description/language
	Update	UpdateSearchSpaces	Update publish flag
			Update description data
Delete	UpdateSearchSpaces	Remove description/language	
LISTPRICE	Add	UpdateSearchSpaces	Add list price
	Update	UpdateSearchSpaces	Update list price
	Delete	UpdateSearchSpaces	Delete list price
ATTRVALUE	Add	UpdateSearchSpaces	Add user-defined value
	Update	UpdateSearchSpaces	Update user-defined value
	Delete	UpdateSearchSpaces	Delete user-defined value
CATENTATTR	Add	UpdateSearchSpaces	Add attribute
	Update	UpdateSearchSpaces	Update attribute
	Delete	UpdateSearchSpaces	Delete attribute
CATGPENREL	Add	AddProductsToSearchSpace	Add product to category
	Delete	RemoveProductsFromSearchSpace	Remove product from category

See Chapter 6, “Loading data” on page 41 for more information.

- **Extractor**

The Extractor uses a query against a database to extract selected subsets of data from the database into an XML document. The data to be extracted from the database is specified using an extraction-filter XML document.

The function of the Extractor is opposite that of the Loader. You use the Extractor to extract selective subsets of data from the WebSphere Commerce database in the form of XML files. You can extract data on products related to an upcoming holiday, for example, or you can extract information from a consolidated database for use with other systems.

See Chapter 7, “Extracting data” on page 59 for more information.

The Loader package also has a logger feature. Each utility in the Loader package creates messages to indicate success, failure, and errors as well as to provide program trace information.

Administrative tools

The Catalog Manager includes the following tools to assist in the administration of its functions:

Text Transformation tool

The Text Transformation tool helps the administrator to prepare the information needed to process a transformation of data between a character-delimited variable format and an XML data format.

XSL Editor

The XML Transformer uses Extensible Stylesheet Language (XSL) to define the rules for transforming an XML file into another XML file. The mapping function in the XSL Editor gives the administrator a visual interface with which to establish the association from an element in a source document type definition (DTD) to an element in a target DTD.

Chapter 2. Transforming text

The Catalog Manager gives you the ability to convert ASCII-file output from other tools (such as spreadsheet programs) into an XML data format that can be entered into the WebSphere Commerce database.

The Text Transformation tool prepares the information needed to process a transformation of data between a character-delimited variable format and an XML data format. The following views are provided:

1. The Text Schema Edit View allows you to create and modify the XML schema file to be used in a transformation.
2. The Transformation Command Edit View allows you to create and modify the actual commands used to run the transformation process.
3. The Transformation Command Process View allows you to launch the transformation process.

Launching the Text Transformation tool

To launch the Text Transformation tool, use the appropriate script or command provided in the WebSphere Commerce directory:

- ▶ **NT** `drive:\WebSphere\CommerceServer\bin\TextTrans.cmd`
- ▶ **2000** `drive:\Program Files\WebSphere\CommerceServer\bin\TextTrans.cmd`
- ▶ **AIX** `/usr/WebSphere/CommerceServer/bin/TextTrans.sh`
- ▶ **Solaris** ▶ **Linux** `/opt/WebSphere/CommerceServer/bin/TextTrans.sh`

▶ **400** In iSeries environments, administrators should first copy the files necessary in order to run the Text Transformation tool to their Windows NT or Windows 2000 machines. The following is an example of how to do this:

1. Create a new directory on the Windows machine (`drive:\TextTrans` for example).
2. Create the following new subdirectories under `drive:\TextTrans`:

```
\bin
\lib\loader
\wcsadmin
```

3. Copy the files from the iSeries machine to these directories according to the following chart:

From		To
/QIBM/ProdData/WebCommerce		drive:\TextTrans
/bin		\bin
/TextTrans.cmd	----->	\TextTrans.cmd
/lib/loader		\lib\loader
/wcmxmlp.jar	----->	\wcmxmlp.jar
/wcmxslt.jar	----->	\wcmxslt.jar
/wcsadmin		\wcsadmin
/acsxml.ico	----->	\acsxml.ico
/swing.jar	----->	\swing.jar
/TextTransformerUI.zip	----->	\TextTransformerUI.zip
/TextTransform.cnt	----->	\TextTransform.cnt
/TextTransform.hlp	----->	\TextTransform.hlp
/TextTransform.reg	----->	\TextTransform.reg

4. Modify the TextTrans.cmd file on the Windows machine by inserting the following text:

```
set WCS_HOME=drive:\TextTrans
```

before the following:

```
set lib=%WCS_HOME%\lib\loader
```

5. Launch the Text Transformation tool by running TextTrans.cmd on the Windows machine.

Using the Text Schema Edit View

The following procedures are related to creating and modifying an XML schema file for transforming data between a character-delimited variable format and an XML format.

Creating a schema file

To create a new schema file, do the following:

1. Launch the Text Transformation tool.
2. Select **File > New**, or click the New icon on the toolbar.
3. Select a path, and type the name of the XML schema file to create.

Note: The default file name is "Default.xml."

4. Select **CSV Format**.

Note: Do not select **WebSphere Commerce Suite Format**. This option is only used with earlier versions of WebSphere Commerce.

5. Click **OK** when you are finished.

Now you can create new elements and attributes by following the steps indicated below.

Opening a schema file

To open a schema file, do the following:

1. Launch the Text Transformation tool.
2. Select **File > Open**, or click the Open icon on the toolbar.
3. Select a schema file to be opened.
4. Click **OK** when you are finished.

Now you can modify the elements and attributes by following the steps described below.

Saving a schema file

To save a schema file, do the following:

1. Launch the Text Transformation tool.
2. To save any changes that you make to the schema, select **File > Save** or click the Save icon on the toolbar.
3. To save a copy of the schema with a new name, do the following:
 - a. Select **File > Save As**.
 - b. Select a path and type the name of the XML file to create.
 - c. Click **OK** when you are finished.

Editing a schema file

To edit a schema file, open the schema file as described above and follow these procedures.

Adding an element

To add an element, do the following:

1. Type a new element name in the Element List field.
2. Click the Add row icon.

Removing an element

To remove an element, do the following:

1. Select an element name.
2. Click the Remove row icon.

Replacing an element

To replace an element, do the following:

1. Type a new element name in the Element List field.
2. Select the name of the element to be replaced.
3. Click the Replace row icon.

Moving an element one row up

To move an element one row up, do the following:

1. Select the name of the element to be moved up.
2. Click the Move row up icon.

Moving an element one row down

To move an element one row down, do the following:

1. Select the name of the element to be moved down.
2. Click the Move row down icon.

Adding an attribute

To add an attribute, do the following:

1. Type a new attribute name in the Attribute List field.
2. Click the Add row icon.

Removing an attribute

To remove an attribute, do the following:

1. Select an attribute name.
2. Click the Remove row icon.

Replacing an attribute

To replace an attribute, do the following:

1. Type a new attribute name in the Attribute List field.
2. Select the name of the attribute to be replaced.
3. Click the Replace row icon.

Moving an attribute one row up

To move an attribute one row up, do the following:

1. Select the name of the attribute to be moved up.
2. Click the Move row up icon.

Moving an attribute one row down

To move an attribute one row down, do the following:

1. Select the name of the attribute to be moved down.
2. Click the Move row down icon.

Changing a schema file structure

The File Structure view in the lower pane of the Text Schema Edit View describes the layout of a character-delimited variable file. The following fields are required in the expected file structure:

Field Separator:

Specifies the delimiter that separates attribute values. The default is a comma (",").

Record Separator:

Specifies the delimiter that separates data records. The default is "
" (This is equivalent to an entity reference to \r\n.)

String Delimiter:

Specifies the delimiter that indicates the string start and end boundary. The default is a single quote (" ' ").

Header Included:

A Boolean value, specified as "true" if there is a header line in the text data file and "false" if there is no header line in the text data file. If there is a header line, it must conform to the XML rules for a tag name because the header will be used as the XML tag name in this case. The default is "false."

Number of header lines:

Specifies the line number header lines that exist in the text data file. The default is a zero ("0").

Preparing a schema file to transform data from an XML to a character-delimited variable format

If you are using an XML schema file that was created using the Text Schema Edit View to transform data from an XML format to a character-delimited variable format, you must use a text editor to change the datatype specification in the schema file from "CSV Format" to "XML Format" before processing the transformation.

Using the Transformation Command Edit View

Using the Transformation Command Edit View, you can create a new command file, open an existing command file, or save changes to a command file. The default command file is named "Manifest.txt."

You can create a new command, remove a command, replace a command with edited information, or change the order of a command.

Note: The command file is automatically saved whenever the commands table is updated.

Creating a command

To create a new command, do the following:

1. Specify the source file, either a character-separated variable format file (with a .csv extension) or an XML format file (with a .xml extension).
2. Specify the XML schema file to be used in the transformation.
3. Specify the name of the output file to be created or modified during the transformation process (that is, where the new data will be stored), either an XML format file (with a .xml extension) or a character-separated variable format file (with a .csv extension).
4. Specify a command mode.
Select **Create** if the output file is to be created or **Append** if the output data is to be appended to an existing data file.
5. Click the Add row icon.

Removing a command

To remove a command, do the following:

1. Select a command.
2. Click the Remove row icon.

Editing or replacing a command

To edit or replace a command, do the following:

1. Click the Edit command icon.
The row data is filled in for the appropriate input fields.
2. Change the text in the appropriate input fields.
3. Click the Replace row icon to update the row.

Moving a command one row up

To move a command one row up, do the following:

1. Select a command.
2. Click the Move row up icon.

Note: This changes the transformation-process sequence.

Moving a command one row down

To move a command one row down, do the following:

1. Select a command.
2. Click the Move row down icon.

Note: This changes the transformation-process sequence.

Clearing a command

To clear a command, do the following:

1. Select a command.
2. Click the Clear input fields icon.

This clears the text in the source-file, schema-file, and output-file fields.

Using the Transformation Process View

To launch the text-transformation process, do the following:

1. Type or browse to the name of the parameter file in the File field.
2. Click **Process**.

The output area below the Process button shows information indicating the status of the transformation process. You can save the output information by clicking the Save button at the bottom of the text area, or you can clear all status information by clicking the Clear button.

Chapter 3. Transforming XML data

The Extensible Stylesheet Language (XSL) provides the following:

1. Language for specifying formatting for XML documents
2. Language that describes how to transform an XML file into another regularly structured file

The transformation capability of XSL can be used to transform an XML file into another XML file that conforms to a different XML schema or DTD.

To transform an XML file into an alternate XML format, you must specify the rules for the transformation using a transform XSL rule file.

The following example transforms the data in `MemberSubsystemExtracted.xml` using `MemberSubsystem.xsl` as the transform XSL rule file with Japanese as the national language:

- 

```
java com.ibm.wca.XMLTransformer.XMLTransformer -infile MemberSubsystemExtracted.xml  
-transform MemberSubsystem.xsl -outfile TransMbrStr.xml -param 'language="-10"'
```

- 

```
QWEBCOMM/TRNWCSXML INFILE(MemberSubsystemExtracted.xml)  
TRANSFORM(MemberSubsystem.xsl) INSTROOT(/QIBM/UserData/WebCommerce/instances/my_inst)  
OUTFILE(TransMbrStr.xml) PARAM('language=-10')
```

The XML Transformer uses XSL to define the rules for transforming an XML file into another XML file. The mapping function in the XSL Editor gives you a visual interface with which you can establish the association from an element in a source DTD to an element in a target DTD. Given two DTDs, you can develop XSL rules that determine how an XML file that conforms to the first (source) DTD is transformed into a file that conforms to the second (target) DTD.

Launching the XSL Editor

To launch the XSL Editor, use the appropriate script or command provided in the WebSphere Commerce directory:

- **NT** `drive:\WebSphere\CommerceServer\bin\XSLEditor.cmd`
- **2000** `drive:\Program Files\WebSphere\CommerceServer\bin\XSLEditor.cmd`
- **AIX** `/usr/WebSphere/CommerceServer/bin/XSLEditor.sh`
- **Solaris** **Linux** `/opt/WebSphere/CommerceServer/bin/XSLEditor.sh`

400 In iSeries environments, administrators should first copy the files necessary in order to run the XSL Editor to their Windows NT or Windows 2000 machines. The following is an example of how to do this:

1. Create a new directory on the Windows machine (`drive:\XMLTrans` for example).
2. Create the following new subdirectories under `drive:\TextTrans`:

```
\bin
\lib\loader
\wcsadmin
```

3. Copy the files from the iSeries machine to these directories according to the following chart:

From	To
/QIBM/ProdData/WebCommerce	drive:\XMLTrans
/bin	\bin
/XSLEditor.cmd	\XSLEditor.cmd
/lib/loader	\lib\loader
/wcmxmlp.jar	\wcmxmlp.jar
/wcmxslt.jar	\wcmxslt.jar
/wcsadmin	\wcsadmin
/config.dtd	\acsxml.ico
/swing.jar	\swing.jar
/XML_transform.ico	/XML_transform.ico
/XMLTransformerUI.zip	/XMLTransformerUI.zip
/XMLTransformWP.xml	/XMLTransformWP.xml
/XMLTransformWP.dtd	/XMLTransformWP.dtd
/XMLRuleConfig.xml	/XMLRuleConfig.xml
/XMLTransform.cnt	/XMLTransform.cnt
/XMLTransform.hlp	/XMLTransform.hlp
/XMLTransform.reg	/XMLTransform.reg

4. Modify the `XSLEditor.cmd` file on the Windows machine by inserting the following text:

```
set WCS_HOME=drive:\XMLTrans
```

before the following:

```
set lib=%WCS_HOME%\lib\loader
```

5. Launch the XMLTransformation tool by running `XSLEditor.cmd` on the Windows machine.

Working with mapping rule building areas

When you launch the XSL Editor, it displays the Mapping Rule Building Area window. Use this window to manage your mapping rule building areas.

Creating a mapping rule building area

To create a mapping rule building area, do the following:

1. Launch the XSL Editor.
2. From the drop-down menu, select **[New]**.
3. In the Name field, type a name for the new mapping rule building area.
4. In the Description field, type a short description of the new mapping rule building area.
5. In the Source Schema field, type the name of an existing file or navigate to an existing file to be used as the source schema.
6. In the Target Schema field, type the name of an existing file or navigate to an existing file to be used as the target schema.
7. In the XSL Rule File field, type a name for the new rule file to be created.
You can specify a complete path here. If you do not give a path, the file is created in the current working directory.
8. Click **Open** to create and open the new mapping rule building area.

Modifying a mapping rule building area

To modify a mapping rule building area, do the following:

1. Launch the XSL Editor.
2. Select the mapping rule building area that you want to modify from the drop-down menu.
3. Click **Open** to open the mapping rule building area.
4. Update the fields that you wish to change.
5. Click **Save** to save your changes.

Deleting a mapping rule building area

To delete a mapping rule building area, do the following:

1. Launch the XSL Editor.
2. Select the mapping rule building area that you want to delete from the drop-down menu.
3. Click the Remove button to remove the entry.

Note: Removing a mapping rule building area does not delete the physical files from the disk.

Using the XSL Editor

When you open a mapping rule building area using the XSL Editor, the mapping rule building area is displayed in the XSL Editor main window.

In the XSL Editor main window, the left pane shows a hierarchical view of the source DTD that is labeled "Source Schema." The right pane shows a hierarchical view of the target DTD that is labeled "Target Schema."

Creating a mapping rule

Element-to-element mapping

Select and drag an element from the source hierarchy and drop it onto an element in the target hierarchy. An XSL rule is generated and displayed in the Mapping Rule view located at the bottom of the window.

Here is an example of a generated XSL rule:

```
<xsl:template match="merchant">
  <xsl:element name="MERCHANT">
    </xsl:element>
  </xsl:template>
```

Note: Any required but non-existing ancestor relationships are automatically generated.

Attribute-to-attribute mapping

Select and drag an attribute from the source hierarchy and drop it onto an attribute in the target hierarchy. An XSL rule is generated and displayed in the Mapping Rule view located at the bottom of the window.

Here is an example of a generated XSL rule:

```
<xsl:attribute name="MEADDR1">
  <xsl:apply-templates select="@mecmail1"/>
</xsl:attribute>
```

Note: Any required but non-existing ancestor relationships are automatically generated.

Creating a custom mapping expression

To create a custom mapping expression, first select an element or attribute from the target hierarchy. Then, right-click and select the Create Custom Expression menu. The Create Custom Expression window displays with a list of available Templates and Rule Expressions in two drop-down menus. Complete the custom expression doing the following:

1. Select a template to which the custom expression is to be added.
2. Select a rule expression to be created (**Constant Expression** for example).
3. Type a value in the Value column for each parameter listed in the table, and press **Enter** to commit the value.
4. Click **OK** to complete the creation step; or click **Cancel** to cancel without creating a rule.

The generated XSL rule is based on the custom expression defined in the rule-configuration file (XSLRuleConfig.xml). You can modify the rule-configuration file and add new rules if needed. To make a rule available for use in the Rule Expressions list, set the Visibility attribute for that rule to "true."

Deleting a mapping rule

To delete a mapping rule, do the following:

1. Select a rule from the Mapping Rule table.
2. Right-click, and select **Delete**.

The rule and all its descendents are deleted.

Note: The updated mapping rules and generated XSL rules are persisted automatically.

Processing an XML transformation

To process an XML transformation, do the following:

1. Select **Tools > Transform** to bring up the Process Transform window.
2. Complete the required fields:
 - a. In the Input XML File field, type or browse to the path and name of the source XML data file.
 - b. In the XSL Rule File field, type or browse to the path and name of the mapping-rule file to be used for the transformation.
If a mapping rule building area is open, this field is pre-filled with the mapping-rule file path currently open in the mapping rule building area.
 - c. In the Output XML File field, type or browse to a path and name for the new XML data file to be created during the transformation process.
3. Click **Start** to start the XML transformation process; or click **Close** to exit the window without processing a transformation.

Customizing the Mapping Rule table

To customize the Mapping Rule table, do the following:

1. To hide a column in the table, right-click a cell in the table and select **Hide column**.
2. To show a hidden column in the table, do the following:
 - a. Right-click a cell in the table.
 - b. Select **Show columns** to bring up the list of hidden columns.
 - c. Select the column from the list.

Note: To select multiple columns, press and hold **Shift** then click the column names.
 - d. Click **OK** to show the selected columns or **Cancel** to cancel the operation.
3. To show all hidden columns in table, right-click a cell in the table and select **Show all columns**.

All columns are shown in the default order.

Displaying a complete XSL rule/value expression

From the Value Expression or XSL Rule columns, clicking a cell brings up a window with the completed rule content for the selected row.

Chapter 4. Generating a DTD and schema

The DTD Generator can create a DTD and a schema to use with the Loader package. The DTD Generator uses an input file containing database-table names and generates either a DTD or a DTD and a schema with a detailed XML file describing the database, depending on how you invoke the DTD Generate command.

Setting up the DTD Generator

The Loader DTD maps directly to the WebSphere Commerce database schema. Each table is an element, and each column is an attribute.

Example: Mapping a Loader DTD to a database schema

DDL statements for the CATENTRY table	DTD
CREATE TABLE	<!ELEMENT CATENTRY EMPTY>
"CATENTRY" (<!ATTLIST CATENTRY
"CATENTRY_ID" BIGINT NOT NULL	CATENTRY_ID CDATA #REQUIRED
"MEMBER_ID" BIGINT NOT NULL	MEMBER_ID CDATA #REQUIRED
"CATENTTYPE_ID" CHAR(16) NOT NULL	CATENTTYPE_ID CDATA #REQUIRED
"MARKFORDELETE" INTEGER NOT NULL	MARKFORDELETE CDATA #REQUIRED
"PARTNUMBER" VARCHAR(64) NOT NULL	PARTNUMBER CDATA #REQUIRED
"MFPARTNUMBER" VARCHAR(64)	MFPARTNUMBER CDATA #IMPLIED
"MFNAME" VARCHAR(64)	MFNAME CDATA #IMPLIED
"URL" VARCHAR(254)	URL CDATA #IMPLIED
"FIELD1" INTEGER	FIELD1 CDATA #IMPLIED
"FIELD2" INTEGER	FIELD2 CDATA #IMPLIED
"FIELD3" DECIMAL(20,5)	FIELD3 CDATA #IMPLIED
"FIELD4" VARCHAR(254)	FIELD4 CDATA #IMPLIED
"FIELD5" VARCHAR(254)	FIELD5 CDATA #IMPLIED
"LASTUPDATE" TIMESTAMP	LASTUPDATE CDATA #IMPLIED
"OID" VARCHAR(64)	OID CDATA #IMPLIED
"ONSPECIAL" INTEGER	ONSPECIAL CDATA #IMPLIED
"ONAUCTION" INTEGER	ONAUCTION CDATA #IMPLIED
"BUYABLE" INTEGER	BUYABLE CDATA #IMPLIED
"BASEITEM_ID" INTEGER	BASEITEM_ID CDATA #IMPLIED
"CLASSIFGRP_ID" INTEGER	CLASSIFGRP_ID CDATA #IMPLIED
"ITEMSPC_ID" INTEGER	ITEMSPC_ID CDATA #IMPLIED
"STATE" INTEGER	STATE CDATA "1"
);	>

You can set the way that the DTD Generator functions by doing the following:

1. Create a new DTD Generator customizer property file.

• 

DB2ConnectionCustomizer.properties is located in the DTDGenerator.zip archive. Extract this file, rename it but keep the .properties extension, and place it in a directory that is in the classpath. **Important:** Do not remove or modify the existing DB2ConnectionCustomizer.properties file.

- ▶ 400

`ISeries_GENWCSDTD_Customizer.properties` is located in the `/QIBM/ProdData/WebCommerce/properties` directory. Copy this file to the `/instroot/xml` directory, rename the new file but keep the `.properties` extension, then make any necessary changes to the new file. **Important:** Do not remove or modify the original `ISeries_GENWCSDTD_Customizer.properties` file.

2. Modify the database-driver values in the new file. For example:

```
DBVendorName = DB2
DBDriverName = COM.ibm.db2.jdbc.app.DB2Driver
DBURL = jdbc:db2:
```

where:

- `DBVendorName` is used to select the type of database.
The options are the following:
 - DB2 Universal Database for iSeries (DB2/iSeries)
 - DB2 for other operating systems (DB2)
 - Oracle database (Oracle)
- `DBDriverName` is used to select the JDBC driver.
The options are the following:
 - DB2 Universal Database for iSeries (`com.ibm.db2.jdbc.app.DB2Driver`)
 - DB2 for other operating systems (`COM.ibm.db2.jdbc.app.DB2Driver`)
 - Oracle database (`oracle.jdbc.driver.OracleDriver`)
- `DBURL` is used to specify the URL to access the database.
The options are the following:
 - DB2 Universal Database for iSeries (`jdbc:db2://`)
 - DB2 for other operating systems (`jdbc:db2:`)
 - Oracle database (`jdbc:oracle:oci8:@`)

3. Specify the new file name as the value of the customizer parameter of the DTD Generate command.

Generating a DTD

The `TableNames.txt` input file contains the following database-table names, one on each line:

```
MEMBER
ADDRBOOK
ADDRESS
```

Here is an example of how the DTD Generator can be invoked:

- ▶ NT ▶ 2000 ▶ AIX ▶ Solaris ▶ Linux

```
java com.ibm.wca.DTDGenerator.GenerateDTD -dbname MALL -dbuser db2inst1
-dbpwd db2ibm -outfile wc.dtd -infile TableNames.txt
```

- ▶ 400

```
QWEBCOMM/GENWCSDTD DATABASE(DATABASE_NAME) SCHEMA(MALL)
INSTROOT(/QIBM/UserData/WebCommerce/instances/mser)
PASSWD(mypassword) OUTFILE(wc.dtd) INFILE(TableNames.txt)
```


The output file wc.dtd contains the following:

```

<!ELEMENT MALL (( MEMBER | ADDRBOOK | ADDRESS)*)>
<!ELEMENT MEMBER EMPTY>
<!ATTLIST MEMBER
  MEMBER_ID      CDATA      #REQUIRED
  TYPE           CDATA      #REQUIRED
  STATE         CDATA      #IMPLIED
>
<!ELEMENT ADDRBOOK EMPTY>
<!ATTLIST ADDRBOOK
  ADDRBOOK_ID   CDATA      #REQUIRED
  MEMBER_ID     CDATA      #REQUIRED
  TYPE          CDATA      #IMPLIED
  DISPLAYNAME   CDATA      #REQUIRED
  DESCRIPTION   CDATA      #IMPLIED
>
<!ELEMENT ADDRESS EMPTY>
<!ATTLIST ADDRESS
  ADDRESS_ID     CDATA      #REQUIRED
  ADDRESSTYPE   CDATA      #IMPLIED
  MEMBER_ID     CDATA      #REQUIRED
  ADDRBOOK_ID   CDATA      #REQUIRED
  ORGUNITNAME   CDATA      #IMPLIED
  FIELD3        CDATA      #IMPLIED
  BILLINGCODE   CDATA      #IMPLIED
  BILLINGCODETYPE CDATA      #IMPLIED
  STATUS        CDATA      #IMPLIED
  ORGNAME       CDATA      #IMPLIED
  ISPRIMARY     CDATA      #IMPLIED
  LASTNAME      CDATA      #IMPLIED
  PERSONTITLE   CDATA      #IMPLIED
  FIRSTNAME     CDATA      #IMPLIED
  MIDDLENAME    CDATA      #IMPLIED
  BUSINESSTITLE CDATA      #IMPLIED
  PHONE1        CDATA      #IMPLIED
  FAX1          CDATA      #IMPLIED
  PHONE2        CDATA      #IMPLIED
  ADDRESS1      CDATA      #IMPLIED
  FAX2          CDATA      #IMPLIED
  NICKNAME      CDATA      #REQUIRED
  ADDRESS2      CDATA      #IMPLIED
  ADDRESS3      CDATA      #IMPLIED
  CITY          CDATA      #IMPLIED
  STATE         CDATA      #IMPLIED
  COUNTRY       CDATA      #IMPLIED
  ZIPCODE       CDATA      #IMPLIED
  EMAIL1        CDATA      #IMPLIED
  EMAIL2        CDATA      #IMPLIED
  PHONE1TYPE    CDATA      #IMPLIED
  PHONE2TYPE    CDATA      #IMPLIED
  PUBLISHPHONE1 CDATA      #IMPLIED
  PUBLISHPHONE2 CDATA      #IMPLIED
  BESTCALLINGTIME CDATA      #IMPLIED
  PACKAGESUPPRESSION CDATA      #IMPLIED
  LASTCREATE    CDATA      #IMPLIED
  OFFICEADDRESS CDATA      #IMPLIED
  SELFADDRESS   CDATA      "0"
  FIELD1        CDATA      #IMPLIED
  FIELD2        CDATA      #IMPLIED
  TAXGEOCODE    CDATA      #IMPLIED
  SHIPPINGGEOCODE CDATA      #IMPLIED
>

```

Generating a schema and a detailed XML file

In this example, the DTD Generator is invoked as follows:

- ▶ NT ▶ 2000
java com.ibm.wca.DTDGenerator.GenerateDTD -dbname SAMPLE
-dbuser johndoe -dbpwd password -xmlTableDesc c:\sample\sample.xml
-outfile tables.dtd -tablenames "employee,staff"
- ▶ AIX ▶ Solaris ▶ Linux
java com.ibm.wca.DTDGenerator.GenerateDTD -dbname SAMPLE
-dbuser johndoe -dbpwd password -xmlTableDesc usr/sample/sample.xml
-outfile tables.dtd -tablenames "employee,staff"
- ▶ 400
QWEBCOMM/GENWCSDTD DATABASE(MYDB) SCHEMA(SAMPLE)
INSTROOT(/QIBM/UserData/WebCommerce/instances/mser) PASSWD(mypassword)
OUTFILE(tables.dtd) TABNAMES('employee,staff') XMLTABDESC(/sample/sample.xml)

The schema file is created in the sample directory and has the file name WCAWebForm.xsd. The sample.xml output file contains the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<formList xmlns="WCAWebForm.xsd" dbname="SAMPLE" dtdname="tables.dtd">
  <form name="EMPLOYEE">
    <uniqueIndex name="U2" columns="FIRSTNME, LASTNAME"/>
    <uniqueIndex name="U3" columns="MIDINIT, LASTNAME"/>
    <field name="EMPNO" type="string" maxLength="6" minOccurs='1'
      uniqueKey="true" showColumnInList="true" />
    <field name="FIRSTNME" type="string" maxLength="32" minOccurs='1'
      showColumnInList="true" />
    <field name="MIDINIT" type="string" maxLength="1" minOccurs='1'
      showColumnInList="true" />
    <field name="LASTNAME" type="string" maxLength="15" minOccurs='1'
      showColumnInList="true" />
    <field name="WORKDEPT" type="string" maxLength="3" showColumnInList="true" />
    <field name="PHONENO" type="string" maxLength="4"/>
    <field name="HIREDATE" type="date" maxLength="10"/>
    <field name="JOB" type="string" maxLength="8"/>
    <field name="EDLEVEL" type="integer" maxLength="5" minOccurs='1' />
    <field name="SEX" type="string" maxLength="1"/>
    <field name="BIRTHDATE" type="date" maxLength="10"/>
    <field name="SALARY" type="decimal" maxLength="9"/>
    <field name="BONUS" type="decimal" maxLength="9"/>
    <field name="COMM" type="decimal" maxLength="9"/>
  </form>
  <form name="STAFF">
    <field name="ID" type="integer" maxLength="5" minOccurs='1'
      uniqueKey="true" showColumnInList="true" />
    <field name="NAME" type="string" maxLength="9" showColumnInList="true" />
    <field name="DEPT" type="integer" maxLength="5" showColumnInList="true" />
    <field name="JOB" type="string" maxLength="5" showColumnInList="true" />
    <field name="YEARS" type="integer" maxLength="5" showColumnInList="true" />
    <field name="SALARY" type="decimal" maxLength="7"/>
    <field name="COMM" type="decimal" maxLength="7"/>
  </form>
</formList>
```

Chapter 5. Resolving identifiers

XML data to be loaded into a target database must contain identifiers for XML elements that require them. To generate or locate identifiers for catalog entities in the XML document, invoke the ID Resolve command.

The ID Resolver only resolves identifiers for a primary table. A primary table is one that is listed in the KEYS or SUBKEYS table. If it is necessary to resolve identifiers for a table that is not in KEYS or SUBKEYS, add the table to the SUBKEYS table before running the ID Resolver.

The following are examples of situations in which you may want to use the ID Resolver:

- Loading new content in XML format when identifiers for the data are required
- Updating content when identifiers already exist for an object in the database

The ID Resolver can supply actual identifiers, or identifiers can be resolved using the following techniques:

- Internal-alias resolution
- Properties-file specification
- Unique-index resolution

Setting up the ID Resolver

You can set how the ID Resolver handles timestamps, storage, and database drivers by doing the following:

1. Create a new ID Resolver customizer property file.

- 

DB2ConnectionCustomizer.properties is located in the IdResGen.zip archive. Extract this file, rename it but keep the .properties extension, and place it in a directory that is in the classpath. **Important:** Do not remove or modify the existing DB2ConnectionCustomizer.properties file.

- 

ISeries_RESWCSID_Customizer.properties is located in the /QIBM/ProdData/WebCommerce/properties directory. Copy this file to the /instroot/xml directory, rename the new file but keep the .properties extension, then make any necessary changes to the new file. **Important:** Do not remove or modify the original ISeries_RESWCSID_Customizer.properties file.

2. Modify the values of the properties specified in the new file.
3. Specify the new file name as the value of the customizer parameter of the ID Resolve command.

Setting how the ID Resolver handles timestamps

The following default input-timestamp masks are provided in the ID Resolver customizer property file:

```
InputTimeStampFormat.1 = yyyy-DD hh:mm:ss.SSSSSS
InputTimeStampFormat.2 = yyyy-MM-dd hh:mm:ss.SSSSSS
InputTimeStampFormat.3 = yyyy-DD-hh.mm.ss.SSSSSS
InputTimeStampFormat.4 = yyyy-MM-dd-HH.mm.ss.SSSSSS
InputTimeStampFormat.5 = yyyy-MM-dd-hh.mm.ss.SSSSSS
InputTimeStampFormat.6 = yyyy-MM-dd HH:mm:ss.SSSSSS
InputTimeStampFormat.7 = yyyy-DD HH:mm:ss.SSSSSS
```

You can modify these timestamp masks or add as many masks as you want to your ID Resolver customizer property file. If you add an input timestamp, you must use the next number in the current sequence. (For example, the next input-timestamp mask would be InputTimeStampFormat.8 if you were adding to the above list.)

You can also customize the output timestamp format, microsecond mask, and database-specific format by modifying the values of the following properties in your ID Resolver customizer property file:

```
TargetTimeStampFormat = yyyy-MM-dd HH:mm:ss.SSSSSS
MicroSecondMask = SSSSSS
DatabaseSpecificFormat = YYYY-MM-DD HH24:MI:SS
```

Setting how the ID Resolver handles storage

Here is the section in the ID Resolver customizer property file that specifies default values for the properties relevant to persistent hashmaps:

```
////////////////////////////////////
/// 0 = Normal hashmap with no backend storage
/// 1 = JDBM
////////////////////////////////////

PersistentStorageType = 0

////////////////////////////////////
/// If PersistentStorageType != 0, set MemoryStorageSize to the maximum size
/// of the hashmap in memory data and after that the hashmap will stream
/// the data to a persistent storage as specified
/// If -1, then it uses the normal hashmap with no backend storage
////////////////////////////////////

MemoryStorageSize = 1
```

You can specify how the ID Resolver handles persistent storage by setting a value for PersistentStorageType in your ID Resolver customizer property file.

- If you set PersistentStorageType = 0, the ID Resolver operates in the "normal" manner (where the symbol hashmaps exist in memory).
- If you set PersistentStorageType = 1, JDBM is used to persist the symbols and keys.

You can specify the number of records stored in memory by setting a value for MemoryStorageSize in your ID Resolver customizer property file.

- A value of "1" for MemoryStorageSize indicates that only one record is kept in memory.

- A value of "-1" for MemoryStorageSize has a special significance, indicating that all records are kept in memory.

In this case, the ID Resolver reverts to its "normal" behavior.

Setting how the ID Resolver handles database drivers

The following lines in the ID Resolver customizer property file specify the default values for database drivers:

```
DBVendorName = DB2
DBDriverName = COM.ibm.db2.jdbc.app.DB2Driver
DBURL = jdbc:db2:
```

where:

- DBVendorName is used to select the type of database.
The options are the following:
 - DB2 Universal Database for iSeries (DB2/iSeries)
 - DB2 for other operating systems (DB2)
 - Oracle database (oracle)
- DBDriverName is used to select the JDBC driver.
The options are the following:
 - DB2 Universal Database for iSeries (com.ibm.db2.jdbc.app.DB2Driver)
 - DB2 for other operating systems (COM.ibm.db2.jdbc.app.DB2Driver)
 - Oracle database (oracle.jdbc.driver.OracleDriver)
- DBURL is used to specify the URL to access the database.
The options are the following:
 - DB2 Universal Database for iSeries (jdbc:db2://)
 - DB2 for other operating systems (jdbc:db2:)
 - Oracle database (jdbc:oracle:oci8:@)



Determining how to process data

The ID Resolve command lets you choose the load, update, or mixed method to process the input file.

- Use the load method to process the input file if *all* records in the file *do not exist* in the database.
- Use the update method to process the input file if *all* records in the file *exist* in the database.
- Use the mixed method to process the input file if *only some* records in the file *exist* in the database.

Choosing the load method

The load method for the ID Resolver is used to generate new identifiers for the records that are loaded into the database. With this method, new identifiers are created for the records. The following example is used to generate identifiers for new data:

-  

```
idresgen -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method load -customizer customizer -schemaname mall
```

The load method is the default.

- ▶ **AIX** ▶ **Solaris** ▶ **Linux**

```
./idresgen.sh -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method load -customizer customizer -schemaname mall
```

The load method is the default.

- ▶ **400**

```
QWEBCOMM/RESWC SID DATABASE(DATABASE_NAME) SCHEMA(MALL)
INSTROOT(/QIBM/UserData/WebCommerce/instances/mser)
PASSWD(mypassword) INFILE(input.xml) OUTFILE(output.xml)
METHOD(*LOAD)
```

Choosing the update method

If you specify the update method for the ID Resolver, the records in the input file should already exist in the database. The ID Resolver locates the identifiers in the database. If a record does not exist in the database, the ID Resolver will not be able to resolve the identifier for this record and it will indicate that an error has occurred. The following example is used to locate identifiers for data that already exist in the database:

- ▶ **NT** ▶ **2000**

```
idresgen -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method update -customizer customizer -schemaname mall
```
- ▶ **AIX** ▶ **Solaris** ▶ **Linux**

```
./idresgen.sh -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method update -customizer customizer -schemaname mall
```
- ▶ **400**

```
QWEBCOMM/RESWC SID DATABASE(DATABASE_NAME) SCHEMA(MALL)
INSTROOT(/QIBM/UserData/WebCommerce/instances/mser) PASSWD(mypassword)
INFILE(input.xml) OUTFILE(output.xml) METHOD(*UPD)
```

Choosing the mixed method

If the input data file contains records that already exist in the database as well as some records that are new, the ID Resolver must be run using the mixed method. With this method, the ID Resolver creates new identifiers for records only if the records do not exist in the database. Otherwise, the existing identifier is obtained from the database. The following example is used to generate identifiers for new data and to locate identifiers for data that already exist in the database:

- ▶ **NT** ▶ **2000**

```
idresgen -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method mixed -customizer customizer -schemaname mall
```
- ▶ **AIX** ▶ **Solaris** ▶ **Linux**

```
./idresgen.sh -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method mixed -customizer customizer -schemaname mall
```
- ▶ **400**

```
QWEBCOMM/RESWC SID DATABASE(DATABASE_NAME) SCHEMA(MALL)
INSTROOT(/QIBM/UserData/WebCommerce/instances/mser) PASSWD(mypassword)
INFILE(input.xml) OUTFILE(output.xml) METHOD(*MIX)
```

Using the ID-resolution techniques

The ID Resolver can use a Java properties file to determine which columns of a primary table should be used as lookups for tables that require the identifier of a primary table. A table is primary if it is listed in the KEYS or SUBKEYS table.

To use internal-alias resolution with the ID Resolver, an alias is placed in the primary-key attribute (identifier) in the XML file. The alias can then be used throughout the XML file to refer to that element. This process eliminates the need for a program to determine the unique indexes necessary to build the XML file.

The ID Resolver can also analyze the database schema to determine whether there is a unique index that fulfills its requirements. The ID Resolver looks for a unique index only when there is no entry in the properties file for the table being analyzed or when there is no properties file. If these conditions are true, a unique-index check is performed. The unique index is considered valid if it exists and does not include the primary key for the table.

Specifying a properties file with the ID Resolver

The ID Resolver lets you use an alternate Java properties file to describe which columns of a primary entry should be used as lookups for tables that require the identifier of a primary row.

The default properties file is `IdResolveKeys.properties`, but you can modify it or specify your own file when invoking the ID Resolve command if you wish.

- ▶ NT ▶ 2000 ▶ AIX ▶ Solaris ▶ Linux

`IdResolveKeys.properties` is located in the following directory:

- ▶ NT `drive:\WebSphere\CommerceServer\properties`
- ▶ 2000 `drive:\Program Files\WebSphere\CommerceServer\properties`
- ▶ AIX `/usr/WebSphere/CommerceServer/properties`
- ▶ Solaris ▶ Linux `/opt/WebSphere/CommerceServer/properties`

If you do not place this file in the current directory when you run the ID Resolver, you can place it in a directory defined in the classpath environment variable. You can also specify the full path to this file.

- ▶ 400

To change `IdResolveKeys.properties`, copy it from the `/QIBM/ProdData/WebCommerce/properties` directory, save it to the `/instroot/xml` directory, then make any necessary changes to the new file.

Note: The above directory is in the classpath used by the RESWCSID command.

Using a properties file to generate identifiers

In the following example, you need to resolve identifiers ADDRBOOK_ID and ADDRESS_ID for ADDRBOOK and ADDRESS records respectively. The identifiers for the MEMBER records are already known. Each record requires a valid identifier for the WebSphere Commerce database. In addition, the ADDRBOOK_ID in the ADDRESS record requires the identifier from the primary table to satisfy its foreign-key constraint.

```
<MEMBER
  MEMBER_ID="100"
  TYPE="U"
  STATE="1"
/>
<MEMBER
  MEMBER_ID="101"
  TYPE="U"
  STATE="1"
/>
<ADDRBOOK
  MEMBER_ID="100"
  DISPLAYNAME="Friends"           Actual value of the DISPLAYNAME column
  DESCRIPTION="All my friends"
  TYPE="P"
/>
<ADDRESS
  ADDRBOOK_ID="@Friends"        Refers to the ADDRBOOK using the DISPLAYNAME
                                value as a lookup
  MEMBER_ID="101"
  NICKNAME="Bob"
  ADDRESS1="15 Brave Developers St."
  CITY="Toronto"
  ZIPCODE="A0A0A0"
  COUNTRY="Canada"
  STATUS="P"
/>
```

You need a properties file to identify which columns in the primary row will be used by the relationship rows in resolving the identifier for the foreign-key column. The following procedure ensures that parsing of the above file proceeds correctly.

In `IdResolveKeys.properties`, specify the following:

```
NAMEDELIMITER=@
SELECTDELIMITER=:

ADDRBOOK=@DISPLAYNAME:DISPLAYNAME
ADDRESS=@NICKNAME:NICKNAME
```

`NAMEDELIMITER` and `SELECTDELIMITER` set the delimiters used throughout the properties file, and they must be used consistently.

`ADDRBOOK=@DISPLAYNAME:DISPLAYNAME` states that when an address-book record is received, the identifier for the address-book row is created. The `DISPLAYNAME` field is extracted from the input record and used to form an association to the new identifier. The `DISPLAYNAME` string is used to match the address-book row `DISPLAYNAME` and resolve the identifier needed by the foreign key.

Using the previous input example, in which the DISPLAYNAME is Friends, assume that the identifier created for this record is 12951. The DISPLAYNAME is used as a key look-aside for 12951. Processing continues with the next record, ADDRESS, where ADDRBOOK_ID has the form of "@..." (which indicates that what follows the delimiter is to be used for looking up the address-book identifier). The string matches the DISPLAYNAME, and 12951 is returned and placed in the ADDRBOOK_ID attribute.

```

<MEMBER
  MEMBER_ID="100"
  TYPE="U"
  STATE="1"
/>
<MEMBER
  MEMBER_ID="101"
  TYPE="U"
  STATE="1"
/>
<ADDRBOOK
  ADDRBOOK_ID="12951"
  MEMBER_ID="100"
  DISPLAYNAME="Friends"
  DESCRIPTION="All my friends"
  TYPE="P"
/>
<ADDRESS
  ADDRESS_ID="13051"
  ADDRBOOK_ID="12951"
  MEMBER_ID="101"
  NICKNAME="Bob"
  ADDRESS1="15 Brave Developers St."
  CITY="Toronto"
  ZIPCODE="A0A0A0"
  COUNTRY="Canada"
  STATUS="P"
/>

```

Generated primary key

Value of ADDRBOOK DISPLAYNAME unchanged

Generated primary key

ADDRESS refers to correct ADDRBOOK

Using a properties file with compound keys

A key made up of more than two columns is a compound key. You can define a compound-key lookup in the properties file by specifying both NAMEDELIMITER and SELECTDELIMITER followed by the field names. To have the lookup criteria for ADDRBOOK records be the compound of the display name and the member ID, for example, specifying the following in the properties file:

```
ADDRBOOK=@DISPLAYNAME@MEMBER_ID:DISPLAYNAME MEMBER_ID
```

then the following XML input-file fragment:

```

<ADDRBOOK
  MEMBER_ID="100"
  DISPLAYNAME="Friends"
  DESCRIPTION="All my friends"
  TYPE="P"
/>
<ADDRBOOK
  MEMBER_ID="101"
  DISPLAYNAME="Friends"
  DESCRIPTION="All my friends"
  TYPE="P"
/>

```

ADDRBOOK "Friends" of MEMBER 100

ADDRBOOK "Friends" of MEMBER 101

```

<ADDRESS
  ADDRBOOK_ID="@Friends@100"
  MEMBER_ID="101"
  NICKNAME="Bob"
  ADDRESS1="15 Brave Developers St."
  CITY="Toronto"
  ZIPCODE="A0A0A0"
  COUNTRY="Canada"
  STATUS="P"
/>

```

**Lookup the primary key for ADDRBOOK
"Friends" of MEMBER 100**

would yield the following after resolution:

```

<MEMBER
  MEMBER_ID="100"
  TYPE="U"
  STATE="1"
/>
<MEMBER
  MEMBER_ID="101"
  TYPE="U"
  STATE="1"
/>
<ADDRBOOK
  ADDRBOOK_ID="12951"
  MEMBER_ID="100"
  DISPLAYNAME="Friends"
  DESCRIPTION="All my friends"
  TYPE="P"
/>
<ADDRBOOK
  ADDRBOOK_ID="12952"
  MEMBER_ID="101"
  DISPLAYNAME="Friends"
  DESCRIPTION="All my friends"
  TYPE="P"
/>
<ADDRESS
  ADDRESS_ID="13051"
  ADDRBOOK_ID="12951"
  MEMBER_ID="101"
  NICKNAME="Bob"
  ADDRESS1="15 Brave Developers St."
  CITY="Toronto"
  ZIPCODE="A0A0A0"
  COUNTRY="Canada"
  STATUS="P"
/>

```

ADDRBOOK of interest

ADDRESS refers to correct ADDRBOOK

Using a properties file with cascaded primary keys

The primary table STOREENT defines a primary key STOREENT_ID. STORE, a foreign table referencing STOREENT, defines a primary key STORE_ID that is a foreign key to the primary table STOREENT. This means that the value of STORE_ID must be one of the STOREENT_ID values. STORE_ID, the primary key of the foreign table STORE, therefore has a dual role—primary and foreign.

Let us assume that another table, CONTRACT, is a foreign table on STORE and that the foreign key for CONTRACT, STORE_ID, references the primary key STORE_ID in STORE. The STORE table is therefore a primary table for the CONTRACT table.

Because the STORE table's STORE_ID is referenced from the STOREENT_ID rather than created, the ID Resolver does not create an internal-alias and ID-value association for the STORE table. When the CONTRACT table tries to resolve the STORE_ID from the STORE table, it gets the empty value.

Because of this special condition, you must explicitly specify the creation of the internal alias by creating an entry in the properties file. In `IdResolveKeys.properties`, specify the following:

```
"STORE=@STORE_ID:STORE_ID"
```

This forces the ID Resolver to do the following:

- Create the internal-alias and ID-value association while resolving the STORE_ID as a foreign reference
- Use the association while resolving the STORE_ID for the CONTRACT table

Using the `STORE=@STORE_ID:STORE_ID` entry in the properties file and the following XML input-file fragment:

```
<STOREENT
  IDENTIFIER="Out Fashions"
  MEMBER_ID="-2000"
  STOREENT_ID="@storeent_id_1"
  TYPE="G"
/>
<STORE
  STORE_ID="@storeent_id_1"
  STOREGRP_ID="1"
  STORELEVEL="store_level"
/>
<CONTRACT
  CONTRACT_ID="@contract_id_1"
  STATE="0"
  STORE_ID="@storeent_id_1"
/>
```

would yield the following after resolution:

```
<STOREENT
  IDENTIFIER="Out Fashions"
  MEMBER_ID="-2000"
  STOREENT_ID="10501"
  TYPE="G"
/>
<STORE
  STORE_ID="10501"
  STOREGRP_ID="1"
  STORELEVEL="store_level"
/>
<CONTRACT
  CONTRACT_ID="@contract_id_1"
  STATE="0"
  STORE_ID="10501"
/>
```

Using internal-alias resolution

To use internal-alias resolution with the ID Resolver, an alias is placed in the primary-key attribute (identifier) in the XML file. The alias can then be used throughout the XML file to refer to that element. This process eliminates the need for a program to determine the unique indexes necessary to build the XML file.

Internal aliases must be used consistently throughout the file. If an address-book ID ADDRBOOK_ID is aliased to @addrbook_1, all foreign-key references to that ID in the file must use @addrbook_1. Note that aliases are transient. They are not saved; and they cannot be used in a separate XML file without introducing the aliases again.

Partial example of using internal-alias ID resolution

Before resolution:

```
<MEMBER
  MEMBER_ID="100"
  TYPE="U"
  STATE="1"
/>
<ADDRBOOK
  ADDRBOOK_ID="@addrbook_1"           Alias for ADDRBOOK
  MEMBER_ID="100"
  DISPLAYNAME="Friends"
  DESCRIPTION="All my friends"
  TYPE="P"
/>
<ADDRESS
  ADDRESS_ID="@address_1"             Alias for ADDRESS
  ADDRBOOK_ID="@addrbook_1"         Refers to the alias for ADDRBOOK
  MEMBER_ID="101"
  NICKNAME="Bob"
  ADDRESS1="1 Brave Developer St."
  CITY="Toronto"
  ZIPCODE="A3B0F4"
  COUNTRY="Canada"
  STATUS="P"
/>
```

After resolution:

```
<MEMBER
  MEMBER_ID="100"
  TYPE="U"
  STATE="1"
/>
<ADDRBOOK
  ADDRBOOK_ID="11801"                 Generated primary key
  MEMBER_ID="100"
  DISPLAYNAME="Friends"
  DESCRIPTION="All my friends"
  TYPE="P"
/>
<ADDRESS
  ADDRESS_ID="11901"                 Generated primary key
  ADDRBOOK_ID="11801"               Refers to ADDRBOOK entry
  MEMBER_ID="100"
  NICKNAME="Bob"
  ADDRESS1="1 Brave Developer St."
  CITY="Toronto"
  ZIPCODE="A3B0F4"
  COUNTRY="Canada"
  STATUS="P"
/>
```

Using unique-index resolution

Unique-index resolution, the default behavior of the ID Resolver, is used when there is no entry in the properties file for the table being analyzed or when there is

no properties file. Unique-index resolution uses any of the specified unique indexes on a table as a means of locating the identifier. For example, MEMBER_ID plus IDENTIFIER is a unique index on the CATALOG table and can therefore be used as a resolution point to the primary key CATALOG_ID of the CATALOGDSC table.

To update the contents of the database, you need to know the unique key from the primary table in the database. You can query your database to find this out. For example, a DB2 command for retrieving a unique key can look like the following:

```
db2 describe indexes for table schema.tablename show detail
```

Partial example of unique-index resolution

Before resolution:

```
<MEMBER
  MEMBER_ID="100"
  TYPE="0"
  STATE="1"
/>

<CATALOG
  DESCRIPTION="Winter Catalog"
  IDENTIFIER="WC2001"
  MEMBER_ID="100"
  TPCLEVEL="2"
/>

<CATALOGDSC
  CATALOG_ID="@WC2001@100"
  FULLIMAGE="c:\store\img\wc.gif"
  LANGUAGE_ID="-1"
  LONGDESCRIPTION="2001 Winter Catalog"
  SHORTDESCRIPTION="2001 Winter Catalog"
  NAME="InFashion 2001 Winter Catalog"
  THUMBNAIL="c:\store\img\wc_th.gif"
/>
```

Refers back to catalog "WC2001" of member "100" (Note: The order is important.)

After resolution:

```
<MEMBER
  MEMBER_ID="100"
  TYPE="0"
  STATE="1"
/>

<CATALOG
  CATALOG_ID="10351"
  DESCRIPTION="Winter Catalog"
  IDENTIFIER="WC2001"
  MEMBER_ID="100"
  TPCLEVEL="2"
/>
```

Automatically generated primary key

```
<CATALOGDSC
  CATALOG_ID="10351"
  FULLIMAGE="c:\store\img\wc.gif"
  LANGUAGE_ID="-1"
  LONGDESCRIPTION="2001 Winter Catalog"
  SHORTDESCRIPTION="2001 Winter Catalog"
  NAME="InFashion 2001 Winter Catalog"
  THUMBNAIL="c:\store\img\wc_th.gif"
/>
```

Refers to the correct catalog

Loading data into the MEMBER table

The ID Resolver handles resolution for tables that have identifiers generated for them by the system. This includes any table and column registered in the KEYS or SUBKEYS table. This resolution has two components:

1. Determining if a primary table (that is, a table listed in KEYS or SUBKEYS) exists in the database.

This resolution is based on the contents of the XML data for that element using either unique-index resolution or properties-file specification.

2. Determining if there is a foreign key to a primary table.

This is done with a resolution specification in the foreign-key attribute of the related table.

The MEMBER table is used as a "super class" for the ORGENTITY, MBRGRP, and USER tables. This creates an "is-a" pattern that is useful for maintaining referential integrity when tables have foreign-key constraints to the subtypes of the MEMBER table. Because all MEMBER subtypes share a common base type, however, the identifier must be unique among the subtypes. This means that an ORGENTITY_ID must be unique in the MBRGRP_ID and USER_ID set. To accomplish this, the KEYS table refers to only the ORGENTITY, MBRGRP, and USER tables and specifies mutually exclusive ranges for their identifiers. Each of the subtypes has a primary key; each of these primary keys is also a foreign key to the MEMBER table primary key.

The constraints between MEMBER and its subtypes create a situation where a MEMBER and subtype cannot have a synchronized ID. In order to load the ORGENTITY, MBRGRP, and USER tables into the system, the ID Resolver recognizes the "is-a" pattern and deals with it appropriately. The following XML syntax for the ID Resolver:

```
<ORGENTITY
  ORGENTITY_ID="@orgAlias"
  ORGENTITYNAME="Test Org"
  ORGENTITYTYPE="0">
  <ISA>
    <MEMBER
      TYPE="0"
      STATE="1"
    />
  </ISA>
</ORGENTITY>
```

generates the following:

```
<MEMBER
  MEMBER_ID="12345"
  TYPE="0"
  STATE="1"
/>
<ORGENTITY
  ORGENTITY_ID="12345"           Synchronized with member element
  ORGENTITYNAME="Test Org"
  ORGENTITYTYPE="0"
/>
```

In this way, the ID Resolver handles the <isa> subelement and creates a synchronized identifier.

Creating a foreign relationship using the REFKEYS table

The REFKEYS table is created to represent a foreign relationship between tables that does not already exist in the database. Generally, the database schema describes the foreign relationship by creating a foreign-key declaration that links a column of a table to another table. If the database schema does not have a foreign relationship defined and the identifiers have to be resolved as a foreign key, then do the following:

1. Create a REFKEYS table as shown in the following example DDL:

```
CREATE TABLE "REFKEYS" (  
    "FKTABLE_NAME" CHAR(18) NOT NULL ,  
    "FKCOLUMN_NAME" CHAR(18) NOT NULL ,  
    "TABLENAME" CHAR(18) NOT NULL  
);
```

where:

FKTABLE_NAME is the foreign (or "child") table name

FKCOLUMN_NAME is the foreign column name

TABLENAME is the primary (or "parent") table name

2. Create an entry in the REFKEYS table that describes the required foreign relationship.

Troubleshooting errors

If errors occur while resolving identifiers, refer to the following table:

Error	Method Used	Possible Cause	Possible Solution	
Unresolved primary key	All	The ID Resolver does not resolve the primary key (identifier) in a table that is not specified in either the KEYS or SUBKEYS table.	<p>Add the name of the table for which the primary key is to be resolved to the SUBKEYS table before running the ID Resolver.</p> <p>In addition, make sure that the foreign-key relationship exists in the database schema.</p>	
	Update	The primary key is resolved by querying the database. The database query is generated by using either the properties-file entry information or the unique index for the given table. The properties-file entry has priority.	<p>Make sure that the unique-index information in the input file is correct.</p> <p>You may also want to create or modify the appropriate entry in the properties file to generate the appropriate database query to resolve the primary key.</p>	
Unresolved foreign key	All	The ID Resolver does not resolve the foreign key in a table where the foreign references are to tables that are not specified in either the KEYS or SUBKEYS table.	<p>Add the name of the table referenced to the SUBKEYS table before running the ID Resolver.</p> <p>In addition, make sure that the foreign-key relationship exists in the database schema.</p>	
		A foreign key is resolved by using the internal alias or by querying the database. The database query is made only if the internal alias fails to resolve the foreign key.	The internal alias is generated by using the primary key and by using the properties-file entry.	Make sure that the internal-alias values used as source and target in the input file are correct.
			The database query is generated by using either the properties-file entry information or the unique index for the given table. The properties-file entry has priority.	<p>Make sure that the unique-index information in the input file is correct.</p> <p>You may also want to create or modify the appropriate entry in the properties file to generate the appropriate database query to resolve the primary key.</p>

Chapter 6. Loading data

Before you load data, you must do the following:

1. Generate a DTD and schema for use with the Loader (the first time data is being loaded)

Note: If you are loading data for a store archive and created the XML file using the DTDs provided with the store archive, this step is not necessary.

2. Resolve identifiers (if necessary)

Data must be in XML format with an associated DTD. To load data, invoke the Load command.

Setting up the Loader

The Loader package allows you to set up how the Loader functions by doing the following:

- Ignoring elements in the input file
- Inserting NULL into a column
- Loading timestamps and date data
- Loading current timestamps
- Managing event queues
- Running with different database software and operating systems
- Substituting a component
- Using Product Advisor search-space synchronization

You can customize these features of the Loader by doing the following:

1. Create a new Loader customizer property file.

- 

MassLoadCustomizer.properties is located in the MassLoader.zip archive.

Extract this file, rename it but keep the .properties extension, and place it in a directory that is in the classpath. **Important:** Do not remove or modify the existing MassLoadCustomizer.properties file.

- 

ISeries_LODWCSDTA_Customizer.properties is located in the /QIBM/ProdData/WebCommerce/properties directory. Copy this file to the /instroot/xml directory, rename the new file but keep the .properties extension, then make any necessary changes to the new file. **Important:** Do not remove or modify the original

ISeries_LODWCSDTA_Customizer.properties file.

2. Modify the values of the properties specified in the new Loader customizer property file.
3. Specify the new file name as the value of the customizer parameter of the Load command.

Ignoring elements in the input file

If your input file contains elements that do not map to the target database, you can set the Loader to ignore those elements in the Loader customizer property file. Use `IgnoreElements` to specify elements to ignore, and separate these elements with a semicolon (;). To ignore the import, literals, and `ProductRepository` elements, for example, specify the following in the Loader customizer property file:

```
IgnoreElements = import;literals;ProductRepository
```

Inserting NULL into a column

You can enable the Loader to insert NULL into a column by setting the `EnableNULLCheck` property to "true" in the Loader customizer property file. For example:

```
EnableNULLCheck = true
```

For performance reasons, this feature is disabled by default.

Use the `NULLStringLiteral` property to determine the string representation of a null value within your data. To set the Loader so that the string "-" is used to represent a null value, for example, specify the following property and value in the Loader customizer property file:

```
NULLStringLiteral = -
```

By default, the value of this property is "NULL" (with no quotation marks).

Loading timestamps and date data

The Loader can load data into columns with timestamp and date data types. The data formats for timestamp and date data in the document are determined by patterns that can be customized. The user can edit an existing pattern or add more patterns to the existing list of patterns.

The data for a timestamp or date is checked against the available patterns (masks). The first pattern that matches the data is used to convert the data to the target timestamp format before loading it into the database.

There are two customizable output-timestamp patterns, `TimeStampFormat.JDBC` and `TimeStampFormat.Load`.

1. `TimeStampFormat.JDBC` is used when the Loader uses JDBC connections to perform an operation.

The SQL import and delete methods of the Loader use JDBC connections for updating the database.

2. `TimeStampFormat.Load` is used when the Loader uses the native utilities.

The import and load methods of the Loader use native utilities.

You can customize the timestamp formats by modifying or adding masks in the Loader customizer property file.

The following input-timestamp masks are provided:

```
InputTimeStampFormat.1 = yyyy-DD hh:mm:ss.SSSSSS
InputTimeStampFormat.2 = yyyy-MM-dd hh:mm:ss.SSSSSS
InputTimeStampFormat.3 = yyyy-DD-hh.mm.ss.SSSSSS
InputTimeStampFormat.4 = yyyy-MM-dd-HH.mm.ss.SSSSSS
InputTimeStampFormat.5 = yyyy-MM-dd-hh.mm.ss.SSSSSS
InputTimeStampFormat.6 = yyyy-MM-dd HH:mm:ss.SSSSSS
InputTimeStampFormat.7 = yyyy-DD HH:mm:ss.SSSSSS
```

The default patterns for input-date formats are as follows:

```
InputDateFormat.1 = MM-dd-yyyy
InputDateFormat.2 = yyyy-dd-MM
InputDateFormat.3 = yyyy-MM-dd
InputDateFormat.4 = MM/dd/yyyy
InputDateFormat.5 = yyyy/dd/MM
InputDateFormat.6 = yyyy-DD
```

You can modify these timestamp and date masks or add as many masks as you want. Specify these masks in the Loader customizer property file in the numeric sequence in which you want them to be compared with the input timestamp. If you add an input timestamp, you must use the next number in the current sequence. (For example, the next input-timestamp mask would be `InputTimeStampFormat.8` if you were adding to the above list.)

The patterns for formatting input data to the output for timestamp and date are as follows:

```
TimeStampFormat.JDBC = yyyy-MM-dd hh:mm:ss.SSSSSS
TimeStampFormat.Load = yyyy-MM-dd-hh.mm.ss.SSSSSS

DateFormat.JDBC = yyyy-MM-dd
DateFormat.Load = yyyy-MM-dd
```

In general, output date and timestamp formats are *not* customized.

Loading current timestamps

The Loader can insert values into columns with a timestamp data type based on a reading of the time-of-day clock. For example, the `STARTDATE` and `ENDDATE` of an offer in WebSphere Commerce can have values based on the time at which the offer is inserted into the table. To support this functionality, the Loader package uses the `MLTIME` table to keep the timestamp instances. The schema for this table is as follows:

```
table MLTIME
(
  INSTANCEID BIGINT not null,
  MLTIMESTAMP TIMESTAMP
)
```

You can customize the name of the table and its columns by changing the following properties in the Loader customizer property file:

```
TimeStampTableName = MLTIME
TimeStampIdColumn = INSTANCEID
TimeStampValueColumn = MLTIMESTAMP
```

The input data for specifying current-timestamp values are based on timestamp string patterns. The following masks are used for specifying the durations for the timestamp:

%D for days
%M for months
%Y for years
%H for hours
%m for minutes
%s for seconds

You can customize current-timestamp formats by modifying or adding masks in the Loader customizer property file. The following input masks are provided:

```
InputCurrentTimestampFormat.1 = CURRENT TIMESTAMP
InputCurrentTimestampFormat.2 = CURRENT TIMESTAMP %D DAYS
InputCurrentTimestampFormat.3 = CURRENT TIMESTAMP %D DAYS %M MONTHS
InputCurrentTimestampFormat.4 = CURRENT TIMESTAMP %D DAYS %M MONTHS %Y YEARS
InputCurrentTimestampFormat.5 = CURRENT TIMESTAMP %Y YEARS %M MONTHS %D DAYS
InputCurrentTimestampFormat.6 = SYSDATE
InputCurrentTimestampFormat.7 = ADDDAYS(SYSDATE,%D)
InputCurrentTimestampFormat.8 = ADDDAYS(ADDMONTHS(SYSDATE,%M),%D)
InputCurrentTimestampFormat.9 = ADDDAYS(ADDMONTHS(ADDYEARS(SYSDATE,%Y),%M),%D)
```

Input data for the current timestamp is matched with the specified patterns. If the data matches a specified input pattern, that pattern is used to parse the input data and the Loader converts the data into the appropriate output format before inserting it into the database. New patterns can be added to the above list provided the subscript numbers are ordered sequentially.

There are two target output formats for specifying current timestamps:

1. CurrentTimestampFormat.Load is used when the Loader is operating in load or import mode.
2. CurrentTimestampFormat.JDBC is used when the Loader uses JDBC to insert, update, or delete values in the database.

The default target patterns in the Loader are as follows:


```
CurrentTimestampFormat.Load = CURRENT TIMESTAMP %Y YEARS %M MONTHS %D DAYS
                             %h HOURS %m MINUTES %s SECONDS
CurrentTimestampFormat.JDBC = CURRENT TIMESTAMP %Y YEARS %M MONTHS %D DAYS
                             %h HOURS %m MINUTES %s SECONDS
```

The properties for these patterns can also be customized in the Loader customizer property file. When you customize the CurrentTimestampFormat.Load and CurrentTimestampFormat.JDBC properties, you should make sure that the syntax of the resulting statement is valid for the given database management system.

The CurrentTimestampLiteral property is used by the Loader to make an early determination of whether the value for the timestamp column is in a current-timestamp format, thus avoiding expensive computations to determine that the value is not a string representation of timestamp.

```
CurrentTimestampLiteral = CURRENT TIMESTAMP
```

 The default value for this property for DB2 is CURRENT TIMESTAMP.

 The default value for the Oracle database is SYSDATE.

Example of loading current timestamps

The Loader is given the information below to update the offer with an OFFER_ID of 10123. The start date has a value of "CURRENT TIMESTAMP", and the end date has a value of "CURRENT TIMESTAMP + 14 DAYS."

```
<OFFER
  OFFER_ID="10123"
  STARTDATE="CURRENT TIMESTAMP">
  ENDDATE="CURRENT TIMESTAMP + 14 DAYS"
/>
```

The Loader recognizes that the columns STARTDATE and ENDDATE are of a timestamp data type in the database. Based on the CurrentTimeStampLiteral property, the values are determined to have values specified in the current-timestamp format. The value for STARTDATE matches the InputCurrentTimeStampFormat.1 pattern, and it is converted to the pattern specified by the CurrentTimeStampFormat.JDBC property. The value for ENDDATE matches the format of the InputCurrentTimeStampFormat.2 property, and it is also converted to the pattern specified by the CurrentTimeStampFormat.JDBC property.

Examples of adding durations to current timestamps

The Loader gives you the ability to add durations to current timestamps. For example, you may want to load an offer without inputting a specific date. To do that, you must create an end date that is some duration after the start date. The following example works well with DB2:

```
<Offer
  Startdate="Current Timestamp"
  Enddate="Current Timestamp +14 Days +4 Months +1 Year +0 Hours
    +0 Minutes +0 Seconds"
/>
```

To handle current-timestamp durations in a platform-independent way, however, you must customize the current-timestamp formats by modifying the masks in the Loader customizer property file. Here is an example of customized current-timestamp property specifications:

```
CurrentTimeStampLiteral=Current Timestamp

InputCurrentTimeStampFormat.0=Current Timestamp
InputCurrentTimeStampFormat.1=Current Timestamp %D Days
InputCurrentTimeStampFormat.2=Current Timestamp %M Months
InputCurrentTimeStampFormat.3=Current Timestamp %Y Years
InputCurrentTimeStampFormat.4=Current Timestamp %D Days %M Months
InputCurrentTimeStampFormat.5=Current Timestamp %D Days %M Months %Y Years
InputCurrentTimeStampFormat.5=Current Timestamp %H Hours %m Minutes %s Seconds

CurrentTimeStampFormat.JDBC=Current Timestamp %D Days %M Months %Y Years
%H Hours %m Minutes %s Seconds
```

Using the offer example and these property specifications, the end date for the offer matches the InputCurrentTimeStampFormat.5 pattern. This yields the following offer information using the CurrentTimeStampFormat.JDBC.

```
<Offer
  Startdate="Current Timestamp"
  Enddate="Current Timestamp +14 Days +4 Months +1 Year +0 Hours +0 Minutes +0 Seconds"
/>
```

The above example shows how the Loader can input multiple current-timestamp formats and format them appropriately to a desired output format. The following example shows how you can handle platform-independent formats and map them to platform-specific output formats.

```
<Offer
  Startdate="Now"
  Enddate="Now +14D +4M +1Y"
/>

CurrentTimestampLiteral=Now

InputCurrentTimestampFormat.0=Now
InputCurrentTimestampFormat.1=Now %DD
InputCurrentTimestampFormat.2=Now %MM
InputCurrentTimestampFormat.3=Now %YY
InputCurrentTimestampFormat.4=Now %DD %MM
InputCurrentTimestampFormat.5=Now %DD %MM %YY
InputCurrentTimestampFormat.5=Sysdate %HH %mm %ss

CurrentTimestampFormat.JDBC=AddYears (AddMonths (AddDays (AddHours (AddMinutes (AddSeconds
  (Sysdate,%s),%m),%H),%D),%M),%Y)
```

Note: The above statement is just an example. It is used merely to illustrate the customization feature for a hypothetical database management system. It is not valid for DB2 or an Oracle database.

Using the offer example and these property specifications, the end date for the offer will match the InputCurrentTimestampFormat.5 pattern. This yields the following offer information using the CurrentTimestampFormat.JDBC.

```
<Offer
  Startdate="Current Timestamp"
  Enddate="AddYears (AddMonths (AddDays (AddMinutes (AddSeconds (Sysdate,0),0),0),14),4),1)"
/>
```

Managing event queues

You can customize the Loader's event queues by modifying the settings in the Loader customizer property file. For example:

```
QueueLowCount = 35
QueueHighCount = 90
```

The source of the events filling the queue is blocked when the number of elements in the queue reaches the higher limit, preventing more events from queuing up. The queue begins accepting events again when the number of elements in the queue falls below the lower limit.

Running with different database software and operating systems

You can customize the Loader to run with different database software and operating systems by modifying the parameters for the following elements in the Loader customizer property file to specify different database software and operating systems:

- Database-connection command
- Database load-table command
- Database-import command
- System command that invokes loading

To customize one of these items, remove the double-slash comment characters (//) that precede the command in the Loader customizer property file and modify the defaults.

Database-connection command: You can change the parameters of the database-connection command if you wish to modify the defaults (which assume that you are using DB2).

```
DBConnectCommand = connect to {0} user {1} using {2};
```

where:

0 = database name
1 = database user
2 = user password

Database load-table command: You can change the parameters of the database load-table command if you wish to modify the defaults.

```
DBLoadTableCommand = load from {0} of del modified by coldel{1}  
chardel{2} insert into {3} ({4});
```

where:

0 = file name
1 = column delimiter
2 = character delimiter
3 = table name
4 = column names, separated by commas (,)

Database-import command: You can change the parameters of the database-import command if you wish to modify the defaults.

```
DBImportCommand = import from {0} of del modified by coldel{1} chardel{2}  
insert_update into {3} ({4});
```

where:

0 = file name
1 = column delimiter
2 = character delimiter
3 = table name
4 = column names, separated by commas(,)

System command that invokes loading: You can change the parameters of the system command that invokes loading if you wish to modify the defaults. This command runs the native load and import scripts generated by the Loader.

- ▶ 

```
DBLoadCommand = db2c1pex DB2 -z {0} -astvf {1}
```

where:

0 = log file name
1 = command file name

For DB2 running on AIX, for example, the value for the DBLoadCommand property is as follows:

```
db2 -tvf {1} -z {0}
```

- ▶ AIX ▶ Solaris ▶ Oracle

```
DBLoadCommand = sqlldr log={0} control={1} USERID={2}
```

where:

0 = log file name
1 = command file name
2 = database user name

Use the following settings for the various database and operating-system combinations:

- ▶ NT ▶ 2000 ▶ DB2

Set the classpath system-environment variable to include db2/dbconnect.zip for DB2 running on Windows NT or Windows 2000 with the sqlimport, load, import, or delete method.

- ▶ AIX ▶ Solaris ▶ Linux ▶ DB2

For DB2 running in AIX, Solaris, or Linux environments, do the following:

- With the sqlimport, load, import, or delete method, set the classpath system-environment variable to include db2/dbconnect.zip.
- With the load or import method, modify the following properties in the Loader customizer property file.

```
/**  
 * Connection command. (Default is for DB2)  
 * parameter 0 = dbName  
 * parameter 1 = dbUser  
 * parameter 2 = userPasswd  
 */
```

```
DBConnectCommand = connect to {0} user {1} using {2};
```

```
/**  
 * Load Data into Table command. (Default is for DB2)  
 * parameter 0 = filename  
 * parameter 1 = column delimiter  
 * parameter 2 = character delimiter  
 * parameter 3 = name of the table  
 * parameter 4 = name of the columns, separated by comma(,)  
 */
```

```
DBLoadTableCommand = load from {0} of del modified by coldel{1}  
insert into {3} ({4});
```

```
/**  
 * Insert Data into Table command. (Default is for DB2)  
 * parameter 0 = filename  
 * parameter 1 = column delimiter  
 * parameter 2 = character delimiter  
 */
```



```

    * parameter 3 = name of the table
    * parameter 4 = name of the columns, separated by comma(,)s
*/

DBUpdateTableCommand = import from {0} of del modified by coldel{1}
insert_update into {3} ({4});

/**
 * System command to invoke load (Default is for DB2)
 * parameter 0 = logFileName
 * parameter 1 = commandFileName
*/

DBLoadCommand = db2 -z {0} -tf {1}

```

- 
 Modify the following properties in the Loader customizer property file for DB2 running on iSeries with the sqlimport, load, import, or delete method:

```

/**
 * The connect string.
*/

ConnectionStringID = jdbc:db2://

/**
 * The JDBC driver information.
*/

JDBCDriverName = com.ibm.db2.jdbc.app.DB2Driver
DbVendorName=DB2/iSeries

/**
 * Custom writer for load/import methods.
*/

WriterName=com.ibm.wca.MassLoader.Writer.ISeriesWriter

```

- 





Set the classpath system-environment variable to include oracle/dbconnect.zip for the Oracle database running in Windows NT, Windows 2000, AIX, or Solaris environments with the sqlimport, load, import, or delete method.

Substituting a component

You can substitute for a Loader component by giving the following elements in the Loader customizer property file the values of classes that you want to substitute for the default implementations:

ParserName

Name of the parser to be used

ValidatorName

Name of the validator to be used

FormatterName

Name of the formatter to be used

JDBCFormatterName

Name of the formatter when the SQL import method is used

WriterName

Name of the writer to be used

JDBCWriterName

Name of the writer when the SQL import method is used

To replace the Loader's default writer (DefaultWriter) with the writer com.abc.writer.SpecialWriter, for example, specify the following in the Loader customizer property file:

```
WriterName = com.abc.writer.SpecialWriter
```

The Loader will use "com.abc.writer.SpecialWriter" to perform the write function.

Using Product Advisor search-space synchronization

To use Product Advisor search-space synchronization, do the following.

1. Create an XML configuration-information file for the synchronization named "PASyncInfo.xml."
2. In PASyncInfo.xml, specify PASync.xsd as the XML schema to be used. For example:

```
<PASync
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation='PASync.xsd'
```

The PASync.xsd file is provided. The following text shows the contents of PASync.xsd:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd='http://www.w3.org/2001/XMLSchema'>

  <xsd:element name="PASync">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="SearchScheme" />
        <xsd:element ref="Command" minOccurs="1" maxOccurs="unbounded" />
      </xsd:sequence>
      <xsd:attribute name = "member" type="xsd:string" use="required" />
      <xsd:attribute name = "store" type="xsd:string" use="required" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="SearchScheme">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="RelatedTable" />
        <xsd:element ref="Search" minOccurs="1" maxOccurs="unbounded" />
      </xsd:sequence>
      <xsd:attribute name = "tableName" type="xsd:string" use="required" />
      <xsd:attribute name = "primary" type="xsd:string" use="required" />
      <xsd:attribute name = "colName" type="xsd:string" use="required" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="RelatedTable">
    <xsd:complexType>
      <xsd:attribute name = "tableName" type="xsd:string" use="required" />
      <xsd:attribute name = "from" type="xsd:string" use="required" />
      <xsd:attribute name = "to" type="xsd:string" use="required" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="Search">
    <xsd:complexType>
      <xsd:attribute name = "value" type="xsd:string" use="required" />
    </xsd:complexType>
  </xsd:element>

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

```

        <xsd:attribute name = "tableName" type="xsd:string" use="required" />
        <xsd:attribute name = "idColumnName" type="xsd:string" use="required" />
        <xsd:attribute name = "addCommand" type="xsd:string" />
        <xsd:attribute name = "updateCommand" type="xsd:string" />
        <xsd:attribute name = "deleteCommand" type="xsd:string" />
    </xsd:complexType>
</xsd:element>

```

```
</xsd:schema>
```

3. In `PASyncInfo.xml`, specify the member ID and store-entity ID for which the synchronization needs to be done. For example:

```

member = "-2000"
store = "10351"

```

4. Under the search-scheme element in `PASyncInfo.xml`, specify the CATGROUP identifiers that make up the search space. For example:

```

<SearchScheme
  tableName = "catgroup"
  primary = "CATGROUP_ID"
  colName = "identifier" >

  <RelatedTable
    tableName = "catgpenrel"
    from = "CATGROUP_ID"
    to = "CATENTRY_ID" />

  <Search value="Pants" />
  <Search value="Shirts" />

```

```
</SearchScheme>
```

"Pants" and "Shirts" are specified in the example. You can specify as many CATGROUP identifiers as you want.

5. In `PASyncInfo.xml`, specify attributes to determine what commands to schedule. For example:

```

<Command tableName = "CATENTRY" idColumnName = "CATENTRY_ID"
  updateCommand = "UpdateSearchSpaces"
  deleteCommand = "RemoveProductsFromAllSearchSpaces"
/>

<Command tableName = "CATENTDESC" idColumnName = "CATENTRY_ID"
  addCommand = "UpdateSearchSpaces"
  updateCommand = "UpdateSearchSpaces"
  deleteCommand = "UpdateSearchSpaces"
/>

<Command tableName = "LISTPRICE" idColumnName = "CATENTRY_ID"
  addCommand = "UpdateSearchSpaces"
  updateCommand = "UpdateSearchSpaces"
  deleteCommand = "UpdateSearchSpaces"
/>

<Command tableName = "ATTRVALUE" idColumnName = "CATENTRY_ID"
  addCommand = "UpdateSearchSpaces"
  updateCommand = "UpdateSearchSpaces"
  deleteCommand = "UpdateSearchSpaces"
/>

<Command tableName = "CATENTATTR" idColumnName = "CATENTRY_ID"
  addCommand = "UpdateSearchSpaces"
  updateCommand = "UpdateSearchSpaces"
  deleteCommand = "UpdateSearchSpaces"
/>

<Command tableName = "CATGPENREL" idColumnName = "CATENTRY_ID"

```

```

        addCommand = "AddProductsToSearchSpace"
        deleteCommand = "RemoveProductsFromSearchSpace"
    />

```

```

</PASync>

```

6. In the new Loader customizer property file, specify the XML configuration-information file. For example:

```

PASyncDocumentURL = PASyncInfo.xml

```

7. In the new Loader customizer property file, enable synchronization. For example:

```

PASyncEnabled = true

```

8. Use either the SQL import or the delete method with the Load command. Here is an example of appropriate XML input to the Loader:

```

<store-asset>

```

```

    <catentry
      CATENTRY_ID="10351"
      MEMBER_ID="-2000"
      PARTNUMBER="000051"
      CATENTTYPE_ID="ProductBean"
      MFPARTNUMBER="m000051"
      MARKFORDELETE="0"
      BUYABLE="1"
    />

```

```

    <catentry
      CATENTRY_ID="10352"
      MEMBER_ID="-2000"
      PARTNUMBER="000052"
      CATENTTYPE_ID="ProductBean"
      MFPARTNUMBER="m000052"
      MARKFORDELETE="0"
      BUYABLE="1"
    />

```

```

    <catentry
      CATENTRY_ID="10353"
      MEMBER_ID="-2000"
      PARTNUMBER="000053"
      CATENTTYPE_ID="ProductBean"
      MFPARTNUMBER="m000053"
      MARKFORDELETE="0"
      BUYABLE="1"
    />

```

```

    <catentry
      CATENTRY_ID="10358"
      MEMBER_ID="-2000"
      PARTNUMBER="000058"
      CATENTTYPE_ID="ProductBean"
      MFPARTNUMBER="m000058"
      MARKFORDELETE="0"
      BUYABLE="1"
    />

```

```

    <catentry
      CATENTRY_ID="10365"
      MEMBER_ID="-2000"
      PARTNUMBER="000065"
      CATENTTYPE_ID="ProductBean"
      MFPARTNUMBER="m000065"
      MARKFORDELETE="0"
      BUYABLE="1"
    />

```

```

<catentry
  CATENTRY_ID="10372"
  MEMBER_ID="-2000"
  PARTNUMBER="000072"
  CATENTTYPE_ID="ProductBean"
  MFPARTNUMBER="m000072"
  MARKFORDELETE="0"
  BUYABLE="1"
/>

<catgpenrel
  CATGROUP_ID="10354"
  CATENTRY_ID="10372"
  CATALOG_ID="10351"
  SEQUENCE="3"
/>

<catgpenrel
  CATGROUP_ID="10354"
  CATENTRY_ID="10365"
  CATALOG_ID="10351"
  SEQUENCE="4"
/>

<catgpenrel
  CATGROUP_ID="10354"
  CATENTRY_ID="10358"
  CATALOG_ID="10351"
  SEQUENCE="5"
/>

<catgpenrel
  CATGROUP_ID="10355"
  CATENTRY_ID="10372"
  CATALOG_ID="10351"
  SEQUENCE="3"
/>

</store-asset>

```

Note: Disabling Product Advisor search-space synchronization provides better Loader performance; therefore, use this feature only when it is needed.

Customizing Product Advisor search-space synchronization

The Loader package allows you to customize Product Advisor search-space synchronization by modifying the Loader customizer property file to do the following:

- **Enable or disable synchronization**

You can enable or disable synchronization by specifying true or false as the value for the following property in the Loader customizer property file:

```
PASyncEnabled = true
```

- **Specify the configuration-information file for the synchronization**

You can specify which XML configuration-information file the synchronization uses by setting the value for the following property in the Loader customizer property file:

```
PASyncDocumentURL = PASyncInfo.xml
```

- **Specify the schedule query length**

You can specify the schedule query length by setting the value for the following property in the Loader customizer property file:

```
PAScheduleQueryLength = 30
```

The value for this property should be within the range of 20 through 900.

- **Specify the scheduled start time**

You can specify the scheduled start time by specifying an absolute timestamp, a current timestamp, or a current timestamp with duration as the value for the `PAScheduledStartTime` property in the Loader customizer property file.

Note: The format of the timestamp must be appropriate for your database.

Here is an example for DB2 that will run the scheduled job 5 minutes after the load:

```
PAScheduledStartTime = CURRENT TIMESTAMP + 5 MINUTES
```

Oracle Here is an example for the Oracle database that will run the job immediately:

```
PAScheduledStartTime = SYSDATE
```

Determining how to process data when using the Loader

The Loader offers the following options for processing data using the Load command:

- Loading
- Importing
- Using the SQL import feature

Before loading data, you should determine which method of processing would produce the best results.






Choosing the load method

Consider the load method in the following situations:


- If you know that the data is clean, and if the database does not contain any data
- If you know that the data is clean, and if you know the database does not contain the data that is being loaded
- If you know that the data is clean, if the targeted tables do not contain any primary keys, and if you know that the database does not contain the data that is being loaded
- If the load time is your primary concern
- If the database is a local DB2 database

400 With the load method, data is loaded into the database. If the data already exists, the command fails as a result of a duplicate-key error and a duplicate-error message displays.

Choosing the import method

     With the import method for DB2, data is also loaded into the database. If the data already exists, it is not deleted but is updated with new values. Consider this method in any of the following situations:

- If the database management system is DB2
- If you do not know whether the data is clean
- If you have to update large sets of homogeneous data at a column level
- If the load time is not your primary concern
- If the table into which data is imported has primary keys

 With the import method, data is also loaded into the database. If the data already exists, it is not deleted but is updated with new values. Consider this method in any of the following situations:

- If you do not know whether the data is clean
- If the data already exists in the database
- If the load time is not your primary concern
- If the table into which data is imported has primary keys

Choosing the SQL import method

With the SQL import method, JDBC or SQL statements are used to update or insert data into the database. Data is inserted if it does not already exist, and existing data is updated. Consider this method in any of the following situations:

- If you are updating existing data and require column-level updates
There is better error reporting on constraint violations and data-type errors with this method.
- If you know that some of the data is not clean
- If database integrity is your primary concern
- If the database is not local
- If you are using Product Advisor search-space synchronization

Other considerations

• Restrictions on using the load method

The load method cannot insert or update data in bit data fields.

 With the load method, only new records are inserted to the database; existing records are not updated.

 The load method can only be used for local, not remote, DB2 databases.

• Restrictions on using the import method

The database management system must be DB2 in order to use the import method.

The import method cannot insert or update data in bit data fields.

With the import method, the Loader only inserts or updates tables that have primary keys defined on them; the import method cannot insert or update data in tables that do not have a primary key. If the input record only has values for columns that are primary, the record is rejected.

• Comparison of the SQL import and load methods

The SQL import method checks for data consistency, including foreign references, and allows you to update existing data. The load method does not.

- **Comparison of the import and SQL import methods**

The import and SQL import methods perform similar functions. The import method is typically faster, but it requires disk space for temporary files.

The import method can only insert or update tables that have primary keys defined on them; whereas, the SQL import method does not require that tables have primary keys on them.

- **Comparison of methods based on database product used**

The import and load methods use native utilities that are optimized for DB2, while the SQL import method uses JDBC calls (which are generic to many database products).

- **Further considerations**

The delete method is used to delete data that is in the input XML document from the database. The element must contain the values for the primary key or the unique index for the table. If the data being deleted has dependencies to data in another table with "cascade on delete" enabled, the dependent data is also deleted.

If you are using Product Advisor search-space synchronization, you must use the SQL import method for loading data.

Loading large documents

When using the Loader package utilities to load large documents into a database, consider the following items:

- **Java Virtual Machine (JVM) heap size**

By default, the maximum amount of memory allocated to the JVM heap is 64 MB. If this is not increased, the JVM can eventually run out of memory during the load process. The maximum amount of memory allocated to the Java heap can be varied by using the JVM `-mx` option in the Java command.

- **Trace logging**

The trace logger can exhaust the JVM heap when loading a large XML document. Trace information is used mostly for debugging a run if the run fails. If tracing the load process is not necessary, the trace should be turned off. There is a significant performance gain when trace is turned off. The trace is turned off by modifying the logging configuration XML document.

The default logging configuration file is `WCALoggerConfig.xml`. To turn off trace logging, change the trace logger configuration for the Loader from:

```
<logger type="trace">
  <handler type="file">
    <filePath>MassLoadTrace.log</filePath>
    <filter type="Any">
      <messageType name="PUBLIC" />
    </filter>
  </handler>
</logger>
```

to:

```
<logger type="trace">
  <handler type="file">
    <filePath>MassLoadTrace.log</filePath>
    <filter type="Any">
    </filter>
  </handler>
</logger>
```

For more information on modifying the `WCALoggerConfig.xml` file, see “Customizing logging for the Loader package” on page 64.

- **Commit count**

The default commit count for the Loader when it is operating in SQL import mode is 1. By default, therefore, transactions are committed for every update or insert into the database. To improve the performance of the Loader for large documents, the commit count should be increased. A value of “100” is suggested; but it can be higher depending on the amount of physical memory on the server, the DBMS transaction log size, and so forth.

The commit count for the Loader is changed using the `-commitcount` *count* option for the Load command (where *count* is the number of statements executed before the transaction is committed).

Troubleshooting tip

If progress is unusually slow when loading data, the logger for the Loader may have a file handler that is not configured correctly. This could result from one of the following situations:

- The user invoking the Loader does not have permission to write to the directory or to update the file specified in the logging configuration document.
- The directory specified as the location of the file in the logging configuration document does not exist.
- The drive specified as the location of the file in the logging configuration document does not have enough space.

When you correct any of these problems, you may need to change the specified location of the file by modifying the logging configuration document (`WCALoggerConfig.xml` by default). For more information on file handlers and the `WCALoggerConfig.xml` file, see “Customizing logging for the Loader package” on page 64.

Chapter 7. Extracting data

To extract data from a database using the Extractor, you must specify the data that you want to extract from the database using an extraction-filter file. The extraction filter that you use depends on the type of data that you want to extract.

The following example extracts Member subsystem data from a database using MemberSubsystemFilter.xml as an extraction filter.

- ▶ NT ▶ 2000 ▶ AIX ▶ Solaris ▶ Linux
java com.ibm.wca.MassExtract.Extract -filter MemberSubsystemFilter.xml
-outfile MemberSubsystemExtracted.xml -dbname mall -dbuser myname
-dbpwd mypassword -customizer MemberSubsystemCustomizer
- ▶ 400
QWBCOMM/EXTWCSDTA FILTER(MemberSubsystemFilter.xml)
OUTFILE(MemberSubsystemExtracted.xml) DATABASE(database_name)
SCHEMA(mall) INSTRROOT(/QIBM/UserData/WebCommerce/instances/mser)
PASSWD(mypassword)

Creating an extraction filter

The following example of an extraction filter extracts category and product information from the CATGROUP, CATGRPDESC, CATGRPREL, CATENTRY, CATENTSHIP, OFFER, CATENTREL, CATGPENREL, CATENTDESC, and ATTRVALUE tables:

```
<sqlx>
<!-- ***** -->
<!-- extract Category information -->
<!-- ***** -->

<functionDef id="Category" description="Extract Categories" schemaentity="catgroup">
  <paramDef name=":lastRecord" type="string" value="10301" description="Last record
before loading new data" />
  <body>
    select * from catgroup where catgroup_id > :lastRecord
  </body>
</functionDef>

<execute id="Category" description="Extract Categories" schemaentity="catgroup">
  <param name=":lastRecord" type="string" value="10300" description="Last record
before loading new data" />
</execute>

<functionDef id="Category Description" description="Extract Category Descriptions
for a Locale" schemaentity="catgrpdesc">
  <paramDef name=":lastRecord" type="string" value="10300" description="Last record
before loading new data" />
  <body>
    select * from catgrpdesc where catgroup_id > :lastRecord
  </body>
</functionDef>

<execute id="Category Description" description="Extract Category Descriptions
for a Locale" schemaentity="catgrpdesc">
  <param name=":lastRecord" type="string" value="10300" description="Last record
before loading new data" />
</execute>

<functionDef id="Category Relationship" description="Extract Category-Relations
```

```

for a Locale" schemaentity="catgrprel">
  <paramDef name=":lastRecord" type="string" value="10300" description="Last record
  before loading new data" />
  <body>
    select * from catgrprel where catgroup_id_child > :lastRecord
  </body>
</functionDef>

<execute id="Category Relationship" description="Extract Category-Relations for
a Locale" schemaentity="catgrprel">
  <param name=":lastRecord" type="string" value="10300" description="Last record
  before loading new data" />
</execute>

<!-- ***** -->
<!-- extract Product information -->
<!-- ***** -->

<functionDef id="Product" description="Extract Product" schemaentity="catentry">
  <paramDef name=":lastrecord" type="string" value="10300" description="Last record
  before loading new data" />
  <body>
    select * from catentry where catentry_id > :lastrecord
  </body>
</functionDef>

<execute id="Product" description="Extract Product" schemaentity="catentry">
  <param name=":lastrecord" type="string" value="10300" description="Last record
  before loading new data" />
</execute>

<functionDef id="Product Relationship" description="Extract Product Ship
information" schemaentity="catentrel">
  <paramDef name=":lastrecord" type="string" value="10300" description="Last record
  before loading new data" />
  <body>
    select * from catentrel where catentry_id_child > :lastrecord
  </body>
</functionDef>

<execute id="Product Relationship" description="Extract Product Ship information"
schemaentity="catentrel">
  <param name=":lastrecord" type="string" value="10300" description="Last record
  before loading new data" />
</execute>

<functionDef id="Product Description" description="Extract Product Description"
schemaentity="catentdesc">
  <paramDef name=":lastrecord" type="string" value="10300" description="Last record
  before loading new data" />
  <body>
    select * from catentdesc where catentry_id > :lastrecord
  </body>
</functionDef>

<execute id="Product Description" description="Extract Product Description"
schemaentity="catentdesc">
  <param name=":lastrecord" type="string" value="10300" description="Last record
  before loading new data" />
</execute>

<functionDef id="Product Ship" description="Extract Product Ship information"
schemaentity="catentship">
  <paramDef name=":lastrecord" type="string" value="10300" description="Last record
  before loading new data" />
  <body>
    select * from catentship where catentry_id > :lastrecord
  </body>
</functionDef>

<execute id="Product Ship" description="Extract Product Ship information"
schemaentity="catentship">
  <param name=":lastrecord" type="string" value="10300" description="Last record

```

```

    before loading new data" />
</execute>

<functionDef id="Category Product Relationship" description="Extract Category
Product Relations" schemaentity="catgpenrel">
  <paramDef name=":lastrecord" type="string" value="10300" description="Last record
before loading new data" />
  <body>
    select * from catgpenrel where catgroup_id > :lastrecord
  </body>
</functionDef>

<execute id="Category Product Relationship" description="Extract Category Product
Relations" schemaentity="catgpenrel">
  <param name=":lastrecord" type="string" value="10300" description="Last record
before loading new data" />
</execute>

<!-- ***** -->
<!-- Extract Product Attribute Information -->
<!-- ***** -->

<functionDef id="Product Attribute Values" description="Extract Product Attribute
values for a Locale" schemaentity="attrvalue">
  <paramDef name=":lastrecord" type="string" value="10300" description="Last record
before loading new data" />
  <body>
    select * from attrvalue where catentry_id > :lastrecord
  </body>
</functionDef>

<execute id="Product Attribute Values" description="Extract Product Attribute values
for a Locale" schemaentity="attrvalue">
  <param name=":lastrecord" type="string" value="10300" description="Last record
before loading new data" />
</execute>

<!-- ***** -->
<!-- Extract Product Price Information -->
<!-- ***** -->

<functionDef id="Offer" description="Extract Offer" schemaentity="offer">
  <paramDef name=":lastrecord" type="string" value="10300" description="Last record
before loading new data" />
  <body>
    select * from offer where catentry_id > :lastrecord
  </body>
</functionDef>



<execute id="Offer" description="Extract Offer" schemaentity="offer">
  <param name=":lastrecord" type="string" value="10300" description="Last record
before loading new data" />
</execute>

</sqlx>

```

Setting up the Extractor

You can change the database drivers that the Extractor uses by doing the following:

1. Create a new Extractor customizer property file.
 -  DB2ConnectionCustomizer.properties is located in the MassExtract.zip archive. Extract this file, rename it but keep the .properties extension, and place it in a directory that is in the classpath. **Important:** Do not remove or modify the existing DB2ConnectionCustomizer.properties file.
 -  ISeries_EXTWCSDTA_Customizer.properties is located in the /QIBM/ProdData/WebCommerce/properties directory. Copy this file to the /instroot/xml directory, rename the new file but keep the .properties extension, then make any necessary changes to the new file. **Important:** Do not remove or modify the original ISeries_EXTWCSDTA_Customizer.properties file.
2. Modify the database-driver values in the new file.
3. Specify the new file name as the value of the customizer parameter of the Extract command.

The following is an excerpt from the Extractor customizer property file:

```
DBVendorName = DB2
DBDriverName = COM.ibm.db2.jdbc.app.DB2Driver
DBURL = jdbc:db2:
```

where:

- DBVendorName is used to select the type of database.
The options are the following:
 - DB2 Universal Database for iSeries (DB2/iSeries)
 - DB2 for other operating systems (DB2)
 - Oracle database (Oracle)
- DBDriverName is used to select the JDBC driver.
The options are the following:
 - DB2 Universal Database for iSeries (com.ibm.db2.jdbc.app.DB2Driver)
 - DB2 for other operating systems (COM.ibm.db2.jdbc.app.DB2Driver)
 - Oracle database (oracle.jdbc.driver.OracleDriver)
- DBURL is used to specify the URL to access the database.
The options are the following:
 - DB2 Universal Database for iSeries (jdbc:db2://)
 - DB2 for other operating systems (jdbc:db2:)
 - Oracle database (jdbc:oracle:oci8:@)

Chapter 8. Using the Loader package logger

Each utility in the Loader package creates messages to indicate success, failure, and errors as well as to provide program trace information.

The utilities in the Loader package reference the `WCALoggerConfig.xml` file.

- ▶ NT ▶ 2000 ▶ AIX ▶ Solaris ▶ Linux

This file exists in a directory specified in the `classpath` system-environment variable. It can also be specified by the `com.ibm.wca.logging.configFile` Java systems property.

- ▶ 400

This file is located in the `/instroot/xml` directory.

`WCALoggerConfig.xml` determines what logging information each utility provides and where the information is displayed or stored. You can customize this file and specify what types of logs are created as well as what types of messages are logged.

Configuring logging in your environment for Windows NT, Windows 2000, AIX, Linux, and Solaris systems

To set up logging in your environment, you must either set the `classpath` system-environment variable to include the file `WCALoggerConfig.xml` or specify the `com.ibm.wca.logging.configFile` system property.

Example of setting the classpath variable

If the `WCALoggerConfig.xml` file is in the directory `d:\WebSphere\CommerceServer\xml\loader` on a Windows NT machine, for example, you can use the following statement to set the classpath variable:

```
SET CLASSPATH=%CLASSPATH%;D:\WebSphere\CommerceServer\xml\loader
```

Example of specifying the `com.ibm.wca.logging.configFile` system property

To specify the `com.ibm.wca.logging.configFile` system property, use the `-D` option when invoking the Java interpreter. An example follows:

```
java -Dcom.ibm.wca.logging.configFile=D:\ice_tea\src\classlib\logger\xml\WC.xml  
com.ibm.wca.DTDGenerator.GeneratedDTD
```

Customizing logging for the Loader package

To customize logging for the Loader package, use the `WCALoggerConfig.xml` file.

- ▶ NT ▶ 2000 ▶ AIX ▶ Solaris ▶ Linux

This file exists in a directory specified in the classpath system-environment variable. It can also be specified by the `com.ibm.wca.logging.configFile` Java systems property.

- ▶ 400

This file is located in the `/instroot/xml` directory.




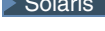

`WCALoggerConfig.xml` contains one or more component tags, `<component name="DTDGenerator">` for example. Within each of these tags, you can add loggers and handlers. You should *not* alter the utility and logger tags provided with the system, but you can add handler tags to the loggers. For information about what you can include in this file, see the `WCALogger.dtd` file.

Loader package logs are located in the `messages.txt` file in the following directories:

- ▶ NT `drive:\WebSphere\CommerceServer\instances\instance_name\logs`
- ▶ 2000 `drive:\Program Files\WebSphere\CommerceServer\instances\instance_name\logs`
- ▶ AIX `/usr/WebSphere/CommerceServer/instances/instance_name/logs`
- ▶ Solaris ▶ Linux `/opt/WebSphere/CommerceServer/instances/instance_name/logs`
- ▶ 400 `/QIBM/UserData/WebCommerce/instances/instance_name/logs`

Handlers

To add a handler to a logger, specify the handler type in the `WCALoggerConfig.xml` file. You can add more than one handler to a logger. Note that each handler has its own attributes and subordinate tags that do not necessarily apply to other handlers. Handler types include the following:

Handler type	Description and attributes
console	Sends messages to standard output, typically the command line
file	Stores messages in a text file You must add "<filePath>log path</filePath>" to this handler as subordinate tagging.
multifile	Creates a circular log of files You must specify "<filePath>log path</filePath>". Log files 1 through <i>n</i> are created. You can add the following attributes: MaxFiles Integer indicating how many log files to use before erasing the first log file MaxKBFileSize Integer indicating the maximum number of kilobytes to store in each log file
     database	Stores messages in a DB2 table in a circular log. You can add the following attributes: brand Database brand name. DB2 is the only database currently supported. maxRows Maximum number of records to store in a table before erasing the oldest entry You can include "<jdbc/>" as a subordinate tag and include the following attributes: url URL used in JDBC to access a database (for example, "jdbc:db2:" <i>wcm</i> ", where " <i>wcm</i> " is the name of the database). The database must exist before you run the utility. table Name of the database table where messages will be logged. It must be created with the following DB2 statement: "CREATE TABLE" <i>tablename</i> (KEY char(13) FOR BIT DATA NOT NULL, COMPONENTNAME VARCHAR(30), ENTRY VARCHAR(2000), PRIMARY KEY(key))" userid Database user name. The user must be assigned permissions to update the table. The following DB2 statement will do this: "GRANT SELECT, UPDATE, INSERT, DELETE ON TABLE" <i>tablename</i> TO USER <i>userid</i> " password Database password for the user name specified

The following example adds a handler of the type "database" to a logger:

```
<handler type="database" brand="DB2" maxRows="50">
  <jdbc url="jdbc:db2:wcm"
    table="wcm.log"
    userid="wasuser"
    password="123456"/>
```

```

<filter type="Any">
  <messageType name="FATAL"/>
  <messageType name="ERROR"/>
  <messageType name="WARNING"/>
</filter>
</handler>

```

Filters

Filters can be added to or removed from handlers to include and exclude message types. If a logger has no filters, no messages are logged. Each filter tag has a subordinate messageType tag that lists the message type, which is typically one of the following:

- INFO
- ERROR
- FATAL
- WARNING

Other message types are listed in the WCALogger.dtd file, but most are generally not used with the Loader package.

Filter types include the following:

Filter type	Description and attributes
Any	Includes in the log file any message flagged as one of the messageType types specified For example, if the messageType list includes ERROR and the application generates an ERROR type message, the message is logged.
All	Requires that a message have all specified messageType type attributes before it is included in a log
Exclude	Logs all messages not specified in the in list of messageType tags

The following example of adding a filter to a handler allows FATAL as well as ERROR message types to be logged and other messages to be ignored:

```

<handler type="file">
  <filter type="Any">
    <messageType name="FATAL"/>
    <messageType name="ERROR"/>
  </filter>
</handler>

```

Formats

You can specify one of two formatter types for message formatting:

Formatter type	Description and attributes
safe (default)	Prevents an exception from being set if a message cannot be found in a properties file This formatter creates a message indicating that the resource is missing.
xml	Formats the message in XML format If a message cannot be found, this formatter also writes a message instead of setting an exception.

Example: WCALoggerConfig.xml and WCALogger.dtd

WCALoggerConfig.xml

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE WCALoggerConfig SYSTEM "WCALogger.dtd">
<WCALoggerConfig>
<component name="MassLoader">
  <logger type="message">
    <handler type="file">
      <filePath>c:\temp\out.txt</filePath>
      <filter type="Any">
        <messageType name="FATAL"/>
        <messageType name="ERROR"/>
        <messageType name="WARNING"/>
        <messageType name="INFO"/>
      </filter>
    </handler>
  </logger>
<logger type="trace">
  <handler type="file">
    <filePath>out2.txt</filePath>
    <filter type="Any">
      <messageType name="PUBLIC"/>
    </filter>
  </handler>
</logger>
<logger type="typedMessage">
  <handler type="file">
    <filePath>tout.txt</filePath>
    <filter type="Any">
      <messageType name="FATAL"/>
      <messageType name="ERROR"/>
      <messageType name="WARNING"/>
      <messageType name="INFO"/>
    </filter>
  </handler>
</logger>
<logger type="progress">
  <handler type="console" format="safe">
    <filter type="Any">
      <messageType name="FATAL"/>
      <messageType name="ERROR"/>
      <messageType name="WARNING"/>
      <messageType name="INFO"/>
    </filter>
  </handler>
</logger>
</component>
<component name="DTDGenerator">
  <logger type="message">
    <handler type="console">
      <filter type="Any">
        <messageType name="FATAL"/>
        <messageType name="ERROR"/>
        <messageType name="WARNING"/>
        <messageType name="INFO"/>
      </filter>
    </handler>
  </logger>
<logger type="trace">
  <handler type="console">
    <filter type="Any">
      <messageType name="FATAL"/>
      <messageType name="ERROR"/>
      <messageType name="WARNING"/>
    </filter>
  </handler>
</logger>
</component>
```

```

        <messageType name="INFO"/>
    </filter>
</handler>
</logger>
</component>
</WCALoggerConfig>

```

WCALogger.dtd

```

<!-- This DTD describes how a WCALoggerConfig XML can be structured.
A WCALoggerConfig XML document is the input configuration file for
the WCALoggerFactory class.
-->

<!ELEMENT WCALoggerConfig (component)+>

<!ELEMENT component (logger)+>
<!ATTLIST component name CDATA #REQUIRED>
<!ELEMENT logger (handler+,messageFile?)>
<!ATTLIST logger type (message | trace | typedMessage | progress) "typedMessage">

<!-- messageFile is an optional default properties files that can be used to
make messages locale specific
-->
<!ELEMENT messageFile (#PCDATA)>
<!ELEMENT handler (filePath?, filter, jdbc?)>
<!ATTLIST handler
type ( file|multiFile|console|error|textArea|database|ejbQueue|queue ) "console">

<!-- maxFiles & maxKBFileSize only applies to the multiFile type of handler
-->
<!-- filePath & encoding applies only when the handler is of type file or
multiFile
-->
<!ATTLIST handler maxFiles CDATA #IMPLIED>
<!ATTLIST handler maxKBFileSize CDATA #IMPLIED>
<!ATTLIST handler encoding CDATA #IMPLIED>
<!ATTLIST handler format (safe | xml) "safe">
<!-- maxRecords & brand are only applicable to database handler type
-->
<!ATTLIST handler maxRecords CDATA #IMPLIED>
<!ATTLIST handler brand (DB2) #IMPLIED>
<!-- the jdbc tag must be present within a database handler type tag
-->
<!ELEMENT jdbc EMPTY>
<!ATTLIST jdbc url CDATA #IMPLIED>
<!ATTLIST jdbc table CDATA #IMPLIED>
<!ATTLIST jdbc userid CDATA #IMPLIED>
<!ATTLIST jdbc password CDATA #IMPLIED>

<!ELEMENT filter (messageType+)>
<!ATTLIST filter type (Any | All | Exclude ) "Any">






<!-- the messageType attribute name is one of these JLog IRecordType
constants
-->
<!ELEMENT messageType EMPTY>
<!ATTLIST messageType name ( NONE | ALL | INFO |
INFORMATION | WARN | WARNING | ERR | ERROR |
FATAL | DEFAULT_MESSAGE | API | CALLBACK |
ENTRY_EXIT | ENTRY | EXIT | ERROR_EXC |
MISC_DATA | OBJ_CREATE | OBJ_DELETE |
PRIVATE | PUBLIC | STATIC | SVC | PERF |
LEVEL1 | LEVEL2 | LEVEL3 ) "ALL">
<!ELEMENT filePath (#PCDATA)>

```

Chapter 9. Using the Loader package error reporter

The Loader and ID Resolver include an error reporter that generates an exception document if there is an error.

By default, the exception document is written to the following directory:

-      the directory where the input document resides
-  `/instroot/logs`

To specify the directory to which the exception document is written, use the Java property `com.ibm.wcm.ErrorReporterDir`. An example for the Loader in a Windows NT environment would begin as follows:

```
java -Dcom.ibm.wcm.ErrorReporterDir=d:\massloaderrors  
com.ibm.wca.MassLoader.MassLoad -dbname . . .
```

Note: The user should have permission to write to the specified directory.

The following is a sample DTD (`store-all-error.dtd`) for the error reporter:

```
<!ENTITY % TABLE "calrule | catentry">  
<!ELEMENT store-asset (error, (%TABLE;)*)>  
<!ELEMENT message (#PCDATA) >  
<!ELEMENT error ( message ) >  
<!ATTLIST error  
    locus          CDATA      #REQUIRED  
    id             CDATA      #REQUIRED  
>  
<!ELEMENT calrule (error)>  
<!ATTLIST calrule  
    identifier      CDATA      #REQUIRED  
    calrule_id     CDATA      #REQUIRED  
    calcode_id     CDATA      #REQUIRED  
    startdate      CDATA      #IMPLIED  
    taxcgr_id      CDATA      #IMPLIED  
    enddate        CDATA      #IMPLIED  
    sequence       CDATA      #REQUIRED  
    combination    CDATA      #REQUIRED  
    calmethod_id   CDATA      #REQUIRED  
    calmethod_id_qfy CDATA      #REQUIRED  
    flags          CDATA      #REQUIRED  
    field1         CDATA      #IMPLIED  
    field2         CDATA      #IMPLIED  
>  
<!ELEMENT catentry (error)>  
<!ATTLIST catentry  
    catentry_id    CDATA      #REQUIRED  
    member_id     CDATA      #REQUIRED  
    catenttype_id  CDATA      #REQUIRED  
    partnumber    CDATA      #IMPLIED  
    mfpartment    CDATA      #IMPLIED  
    mfname        CDATA      #IMPLIED  
    markfordelete CDATA      #REQUIRED  
    url           CDATA      #IMPLIED  
    field1        CDATA      #IMPLIED  
    field2        CDATA      #IMPLIED  
    lastupdate    CDATA      #IMPLIED  
    field3        CDATA      #IMPLIED  
    onspecial     CDATA      #IMPLIED
```

```

onauction      CDATA      #IMPLIED
field4         CDATA      #IMPLIED
field5         CDATA      #IMPLIED
buyable       CDATA      #IMPLIED

```

>

The following is a sample error-report document from the Loader:

```

<?xml version="1.0"?>
<!DOCTYPE store-asset SYSTEM "store-all-error.dtd">
<store-asset>
  <error
    locus="Parser"
    id="SAXParseFatalError" >
    <message>
      Error The string "--" is not permitted within comments. : 155 : 18
    </message>
  </error>
  <calrule
    calcode_id="30"
    enddate="2100-01 10:20:30.000000"
    calmethod_id="-47"
    identifier="7"
    taxcgry_id="9"
    calmethod_id_qfy="-46"
    startdate="1900-01-01-00.00.00.000000"
    flags="1"
    combination="2"
    calrule_id="44"
    sequence="9.0E+1">
    <error
      locus="Writer"
      id="SQLException" >
      <message>
        A SQL Exception was received [IBM][CLI Driver][DB2/NT] SQL0530N
        The insert or update value of the FOREIGN KEY
        "JANTONY.CALRULE.F_CALRULE4" is not equal to any value of the
        parent key of the parent table. SQLSTATE=23503
      </message>
    </error>
  </calrule>
  <catentry
    catentry_id="10118"
    member_id="-2001"
    partnumber="1254"
    mfpartnumber="sku-163"
    mfname="InFashion"
    markfordelete="0"
    buyable="1"
    field1="abc" >
    <error
      locus="Formatter"
      id="FormattingError" >
      <message>
        Error when formatting value for CATENTRY.FIELD1 : abc with error
        [class java.lang.NumberFormatException(abc)].
      </message>
    </error>
  </catentry>
</store-asset>

```

Chapter 10. Configuring Loader package commands and scripts

To launch the Loader package and run its commands, use the scripts or commands provided in the WebSphere Commerce directory:

- ▶ **NT** *drive:\WebSphere\CommerceServer\bin*
- ▶ **2000** *drive:\Program Files\WebSphere\CommerceServer\bin*
- ▶ **AIX** */usr/WebSphere/CommerceServer/bin*
- ▶ **Solaris** ▶ **Linux** */opt/WebSphere/CommerceServer/bin*
- ▶ **400** QWEBCOMM native library

The scripts and commands are as follows:

▶ **400**

GENWCSDTD
DTD Generate command

RESWCSID
ID Resolve command

EXTWCSDTA
Extract command

LODWCSDTA
Load command

TRNWCSTXT
Text Transform command

TRNWCXML
XML Transform command

▶ **NT** ▶ **2000**

dtdgen.cmd
DTD Generate command

idresgen.cmd
ID Resolve command

massextract.cmd
Extract command

massload.cmd
Load command

txttransform.cmd
Text Transform command

xmltransform.cmd
XML Transform command

▶ AIX ▶ Solaris ▶ Linux

dtdgen.sh

DTD Generate shell script

idresgen.sh

ID Resolve shell script

massextract.sh

Extract shell script

massload.sh

Load shell script

txttransform.sh

Text Transform shell script

xmltransform.sh

XML Transform shell script

Note:

The scripts and commands provided in WebSphere® Commerce as utilities for executing the Loader package commands call on scripts and batch files containing environment-variable settings. These are the *setenv.extension*, *setdbenv.db2.extension*, and *setdbenv.oracle.extension* files (where *extension* is the file extension); and they are located in the following directories by default:

- ▶ AIX /usr/WebSphere/CommerceServer/bin
- ▶ 400 QWEBCOMM native library
- ▶ Linux ▶ Solaris /opt/WebSphere/CommerceServer/bin
- ▶ NT drive:\WebSphere\CommerceServer\bin
- ▶ 2000 drive:\Program Files\WebSphere\CommerceServer\bin

You may need to modify these files when you customize your system. On a Windows operating system, for example, you may need to change the values for one or more of the following paths in *setenv.bat*:

- JAVA_HOME
- DB2_DRIVER
- ORACLE_HOME
- ORACLE_DRIVER
- ORACLE_CLASSPATH

These environment variables are set in *setenv.bat* and used in *setdbenv.db2.bat* or *setdbenv.oracle.bat*, depending on the database that you are using.

Part 3. Using the Web editor

This section describes how to administer and use the Catalog Manager Web editor.







The Web editor enables you to create, delete, and make changes to your catalog data through a Web browser. Data-entry forms for viewing and updating information are central to the Web editor. In the simplest case, the forms correspond to tables in the WebSphere Commerce database. The administrator can choose to use the default forms provided or to customize the available forms.

Note: The Web editor uses Internet Explorer 5 and later.

Chapter 11. Setting up the Web editor







The administrator can configure the Web editor to contribute data to WebSphere Commerce, including extensions and customizations to the WebSphere Commerce schema. The Web editor is not designed as a set of Web-entry forms specific to a particular instance of WebSphere Commerce. The Web editor was designed to be flexible and customizable to support the individual needs and roles of different organizations.

A database view is a stored query on one or more tables in the database. During Catalog Manager installation, a sample file for creating a logical view of a WebSphere Commerce product is placed in each system-specific subdirectory (db2, oracle, and os400) of the following directory:

-  *drive:\WebSphere\CommerceServer\schema*
-  *drive:\Program Files\WebSphere\CommerceServer\schema*
-  */usr/WebSphere/CommerceServer/schema*
-   */opt/WebSphere/CommerceServer/schema*
-  */QIBM/ProdData/WebCommerce/schema*

This sample file, *wcs.view.sql*, contains a product view that has been created to combine the information about products from multiple tables. It contains the SQL data definition of the product view. Administrators can study this file to plan for developing their own views of a database.

The default XML form-description file (*forms51_be.xml*) contains forms designed to make it easy to add, edit, and delete data in a WebSphere Commerce database. A copy of this file is located in the following directory:

-  *drive:\WebSphere\CommerceServer\xml\wcwebeditor\xml*
-  *drive:\Program Files\WebSphere\CommerceServer\xml\wcwebeditor\xml*
-  */usr/WebSphere/CommerceServer/xml/wcwebeditor/xml*
-   */opt/WebSphere/CommerceServer/xml/wcwebeditor/xml*
-  */instroot/xml/wcwebeditor/xml*

This configuration file can be used as is, or it can be changed and enhanced by the administrator who sets up the Web editor. To customize the XML form-description file, see the instructions included later in this section.

Note: The DTD Generator can automatically create forms to be used by the Web editor.

Configuring the Web editor

This section provides information on how to configure the Web editor. Although the installation process handles this configuration initially, an administrator can use this information to do things such as reconfiguring the Web editor to use another database.

Editing the `webeditor.properties` file

The Web editor has certain application parameters that are set within the `webeditor.properties` file. It is located in the following directory:

- ▶ **NT** `drive:\WebSphere\AppServer\installedApps\WC_Enterprise_App_demo.ear\wcwebeditor.war\WEB-INF\classes\webeditor.properties`
- ▶ **2000** `drive:\Program Files\WebSphere\AppServer\installedApps\WC_Enterprise_App_demo.ear\wcwebeditor.war\WEB-INF\classes\webeditor.properties`
- ▶ **AIX** `/usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/wcwebeditor.war/WEB-INF/classes/webeditor.properties`
- ▶ **Solaris** ▶ **Linux** `/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/wcwebeditor.war/WEB-INF/classes/webeditor.properties`
- ▶ **400** `/QIBM/UserData/WEBASADV4/was_instance_name/installedApps/WC_Enterprise_App_wcs_instance_name.ear/wcwebeditor.war/WEN-INF/classes`

By editing the `webeditor.properties` file, an administrator can do things such as change which file is used to describe the forms that are displayed in the Web editor. Here is an example of the contents of a `webeditor.properties` file:

```
# Properties file for WebEditor

(The following specifies where the customized process list and the Catalog Manager
utility configuration envelopes are located.)
# URI Location of Process and WCM Subsystem configuration envelopes
ProcessConfigFile=file:///D:/WebSphere/CommerceServer/xml/wcwebeditor/xml/weProcessList.xml

(The following specifies where the forms are defined.)
# Location of Forms file
FormsURL=file:///D:/WebSphere/CommerceServer/xml/wcwebeditor/xml/forms51_be.xml

(The following setting specifies where the XML-to-HTML style sheet is located)
# Location of XML to HTML StyleSheet
StyleSheetURI=file:///D:/WebSphere/CommerceServer/xml/wcwebeditor/xsl/webeditor.xsl

(The following setting specifies the location for temporary files)
# Location of Temporary Directory
#temp.dir=

# WebSphere datasource. This is used to build form drop-down lists and field
# default values. However, when publishing and searching, the Web Editor utilizes
# the WCM subsystems ID Resolver, Mass Loader, and Mass Extractor. Database access
# for these WCM subsystems must be configured separately.
# Name of WAS database source
dbsource=jdbc/WebSphere Commerce DB2 DataSource demo

# Name of WAS database. If specified, this takes preference over the database name
# value in the forms XML file. This value will be utilized as a parameter when
# invoking the WCM subsystems such as the Mass Loader.
dbname=all
```

(The following setting specifies the character set to use. By default, it is

set to the single-byte character set. For other countries, choose from the values that are commented out.)

```
# Encoding Character Set
Encoding=ISO-8859-1
```

```
# CN
#Encoding=gb2312
# TW
#Encoding=Big5
# KR
#Encoding=EUC-KR
# JP
#Encoding=Shift_JIS
```

(The following setting is used to allow images to be previewed. The hostname of the server plus any leading directory information before the image information that is stored in WebSphere Commerce should be specified here. The WebSphere Commerce catalog information is appended to this value to construct an image URL.)

```
# Specifies the base href location for images
# This should be set so that this info plus the info stored in WCS
# (e.g. /image/char.gif) combines to create a URL to an image
```

```
#imageRootURL=http://%HOSTDOMAIN
imageRootURL=http://localhost/webeditor
```

(The following setting is used to set a date format for the application. If these properties are not available (i.e., they are commented out), then the Java-locale specific format will be used. Below is a reference table for setting these values.)

```
# Use these properties to specify a date format if the Java locale-specific
# format is not desired
```

```
#dateFormat=yyyy-MM-dd
#dateTimeFormat=yyyy-MM-dd HH:mm:ss
```

# Symbol	Meaning	Presentation	Example
# G	era designator	(Text)	AD
# y	year	(Number)	1996
# M	month in year	(Text & Number)	July & 07
# d	day in month	(Number)	10
# h	hour in am/pm	(1~12) (Number)	12
# H	hour in day	(0~23) (Number)	0
# m	minute in hour	(Number)	30
# s	second in minute	(Number)	55
# S	millisecond	(Number)	978
# E	day in week	(Text)	Tuesday
# D	day in year	(Number)	189
# F	day of week in month	(Number)	2 (2nd Wed in July)
# w	week in year	(Number)	27
# W	week in month	(Number)	2
# a	am/pm marker	(Text)	PM
# k	hour in day	(1~24) (Number)	24
# K	hour in am/pm	(0~11) (Number)	0
# z	time zone	(Text)	Pacific Standard Time
# '	escape for text	(Delimiter)	
# ''	single quote	(Literal)	'







Changing the location of the temporary files

The location of temporary files can be changed by removing the comment mark and adding a value to the temp.dir property in the webeditor.properties file.

Alternatively, the java.io.tmpdir Java property is used to determine where temporary files are created.

Creating an XML form-description file using the DTD Generator

The forms51_be.xml file is an example of an XML form-description file. It provides a set of forms to be used by the Web editor. A copy of this file is located in the following directory:

-  NT *drive:\WebSphere\CommerceServer\xml\wcwebeditor\xml*
-  2000 *drive:\Program Files\WebSphere\CommerceServer\xml\wcwebeditor\xml*
-  AIX */usr/WebSphere/CommerceServer/xml/wcwebeditor/xml*
-  Solaris  Linux */opt/WebSphere/CommerceServer/xml/wcwebeditor/xml*
-  400 */instroot/xml/wcwebeditor/xml*

The following steps describe how the system administrator can use the DTD Generator to add new XML forms.

Note: Before performing the procedure described below, rename the existing forms51_be.xml in the Web editor directory. Alternatively, create an output file during the following procedure with a new name and then reconfigure the Web editor to use the newly created file. See the previous section for instructions on how to do this.

To create XML forms, run the DTD Generate command.

The following steps describe how to create the XML forms:

1. Create a temporary file named "tables.txt" containing the names of the tables that you want to use in the forms.

Enter each table name on a single line as in the following example:

```
catentry
catentdesc
catentship
inventory
```

2. Save tables.txt to the directory where the DTD Generate command is located. (See Chapter 10, "Configuring Loader package commands and scripts" on page 71 for the installed location of this command.)
3. At the operating-system command prompt, change to the directory where the DTD Generate command is located.

4. Run the DTD Generate command by entering the following:

▶ NT ▶ 2000

```
dttdgen -infile tables.txt -outfile tables51.dtd
-dbname dbname -dbuser userid -dbpwd password
-xmlTableDesc tableFORMS.xml -schemaname schema -propfile filename
```

▶ AIX ▶ Solaris ▶ Linux

```
./dttdgen.sh -infile tables.txt -outfile tables51.dtd
-dbname dbname -dbuser userid -dbpwd password
-xmlTableDesc tableFORMS.xml -schemaname schema -propfile filename
```

▶ 400

```
QWEECOMM/GENWCSDTD DATABASE(database) SCHEMA(schema)
INSTROOT(instroot) PASSWD(password) OUTFILE(tables51.dtd)
INFILE(tables.txt) XMLTABDESC(tableFORMS.xml)
```

The table-description switch (-xmlTableDesc or XMLTABDESC) causes the DTD Generator to create a new form description of the tables in addition to a DTD.

▶ NT ▶ 2000 ▶ AIX ▶ Solaris ▶ Linux

The -propfile option specifies the name of an external properties file where help text, default values, and field-description information can be stored.

5. Reconfigure the Web editor to use the newly created files as described in the previous section.
6. Restart the Web editor in the WebSphere Advanced Administrative Console. To do this, follow these steps:
 - a. Expand **WebSphere Administrative Domain**.
 - b. Expand **Enterprise Applications**.
 - c. Right-click **WebSphere Commerce Enterprise Application - demo**, and select **Stop**.
 - d. Wait until the message indicates that the application has stopped.
 - e. Right-click **WebSphere Commerce Enterprise Application - demo**, and select **Start**.
 - f. Wait until the message indicates that the enterprise application has started.
7. To see the new form in a Web browser, open the following URL:

```
https://host_name:8000/wcm/webeditor
```

where *host_name* is the fully qualified HTTP host name of your WebSphere Application Server.

The Web editor displays in the browser window with a list of all the table names.

Customizing the XML form description

This section describes how the administrator can enhance the forms that the Web editor displays.

The XML form description can be customized and enhanced by setting attributes and values in either the XML form-description file itself or in a separate properties file.

Note: The name of this properties file should be specified as the value of the resourcePackage attribute in the formList tag. If the file name appears in a subdirectory of a directory in the class path, it should use the package(dot) specification.

The following table lists the Web editor form-field attributes that an administrator can alter.

Field attribute	Description
Currency	Causes values to be displayed with a locale-specific numeric separator (such as a comma as a thousands separator for the United States)
DbColumn	Used to map the field name to the properties file key If locale-specific properties files are being used, then the value in this entry must match what is entered in the properties file. The DTD Generator appends a schema to this entry.
DefaultValue	Specifies a value that is shown on the data-entry form when a user fills out a new form This attribute can be set by the DTD Generator to the database default value. It can be a static string, but it can also include an SQL scalar query against a single-line table. To retrieve external data, a user-defined function could be used in the query. For example: DefaultValue="SELECT CURRENT TIMETAMP FROM EXEC" where EXEC is a defined and populated as: CREATE TABLE EXEC (A CHAR(1)); INSERT INTO A VALUES('A');
dynamicSqlSelectionList	Causes a drop-down menu to be rebuilt for each form
FieldDescription	Provides a description that is displayed next to the entry field on the form The DTD Generator uses the comments on the column if there are any when creating this attribute. If there are no comments on the column, then the default is the column name. This attribute can be set in the locale-specific property file or in the XML form-description file. If a value is specified in the properties file, it takes precedence.

Field attribute	Description
FieldHelp	<p>Provides a brief help description of the field to be displayed on the bottom message bar of the browser when the field is in focus on the form.</p> <p>By default, this contains a simple message to enter data for the given field along with the column type of the data.</p> <p>This attribute can be set in the locale-specific property file or in the XML form-description file. If a value is specified in the properties file, it takes precedence.</p>
formatNumber	<p>Used to tell the Web editor not to process a number in any way</p> <p>Set the formatNumber attribute to "false" to treat the value entered as a string except during extract queries (where the value is not placed in quotes as a string would be). The default of this attribute is "true."</p>
Hidden	<p>Indicates that the value is not displayed on the form but is still available as an HTML hidden field</p>
HideOnCreate	<p>Indicates that the field is available when a new form is composed</p> <p>Similar to showInCreateMode="false"; but adds the field name as hidden type.</p>
Maxlength	<p>Specifies the length of the database column</p> <p>It is used to ensure that the user does not enter a value with a length greater than what can be stored in the database.</p>
MinOccurs	<p>Indicates whether the field is required</p> <p>A value of "1" means required, a value of "0" means optional.</p>
Name	<p>Specifies the name of the database column</p>
readOnly readOnlyForCreate readOnlyForEdit	<p>Control when a field is available for editing and when it is only readable by the user</p> <p>readOnly="true" means that a field is always in read-only mode. readOnlyForCreate="true" means that a field is in read-only mode when a new form is being composed. readOnlyForEdit="true" means that a field is in read-only mode when an existing form is being edited.</p>
ShowColumnInList	<p>Specifies that the field is one of the columns that constitute the multi-record view of data when set to "true"</p> <p>The DTD Generator sets the first six columns to "true."</p>

Field attribute	Description
showInCreateMode	Used to hide a field from the created form Set the showInCreateMode attribute to "false" to hide the field on the displayed created form. The default of this attribute is "true."
ShowInSearchMode	Used to hide certain columns from the search-criteria page If this attribute is set to "false," the given field is not displayed on the search form.
SqlSelectionList	Creates a drop-down menu for a user to select from when composing a new form The query must return a one- or two-column result set. The first column is the list of labels that the user can choose from; the second column is the list of actual values to store in the database table. If only one column is used, the value that is displayed is stored in the table. This feature is very helpful in enforcing foreign-key relationships.
Type	Indicates the type of the database column and what kind of validation checks should be performed on the data For example, if the type is "integer," then the application ensures that only a valid integer is entered into this field.
UniqueKey	Indicates that the field is a primary key for a table If this attribute is set to "true," then the application enforces a unique constraint on the data entered for this column. This check is only performed on the records loaded in the application. If does not extend to the records in the database.
ValidateInput	Used to turn off validation checking This attribute allows the user to enter text into a numeric field that will be handled by the ID Resolver or XML Transformer.

Editing form names

The form tag in the XML form description has both name and displayName attributes.

- The name attribute must be set to the name of the database table or view that it represents.
- The displayName attribute provides an easy-to-use name in the Web editor application. Alternatively, a displayName attribute for a form can be set in the properties file specified in the resourcePackage attribute of the formList tag. (The resourcePackage attribute is used for multi-language support.)

Note: If an entry is present in the properties file, it takes precedence over what is entered in the XML form-description file.

Changing a field description

Here is an example of a field description in an XML form-description file:

```
<form name = "CATALOG.CATENTRY"  
  .  
  .  
  <field name="MEMBER_ID"  
    showInCreateMode="false"  
    fieldDescription="MEMBER_ID"  
    type="integer"  
    maxLength="19"  
    defaultValue=""  
    .  
  .  
  <field name="CATENTTYPE_ID"  
    fieldDescription="CATENTTYPE_ID"  
    type="string"  
    maxLength="16"  
    defaultValue=""  
    .  
  .
```

You may want to change it to the following:

```
<form name = "CATALOG.CATENTRY"  
  displayName="Product"  
  .  
  .  
  <field name="MEMBER_ID"  
    showInCreateMode="false"  
    fieldDescription="Member Identifier"  
    type="integer"  
    maxLength="19"  
    defaultValue=""  
    .  
  .  
  <field name="CATENTTYPE_ID"  
    fieldDescription="Product Type"  
    type="string"  
    maxLength="16"  
    defaultValue=""  
    .  
  .
```

Adding a drop-down menu

You can add a selection list in the form of a drop-down menu to a Web editor form to make the form easier to complete and to limit the user to selecting from a valid set of choices.

To refresh the selection lists, the administrator can restart the Web editor. There is also a form-description field attribute, `dynamicSqlSelectionList`, that the administrator can set to "true" to ensure that a drop-down menu reloads each time.

The query for a selection list can have a result set that returns either one or two columns. If the result returns two columns, the second column contains the actual values stored and the first column contains the user labels.

A selection list is created by entering an SQL query in the `sqlSelectionList` attribute of the field tag or by creating an enumeration in the XML form-description file. Both methods are shown in the following example:

```

<field name="MEMBER_ID"
  showInCreateMode="false"
  fieldDescription="Member Identifier"
  .
  .
  readOnly="false"
  sqlSelectionList="select orgentityname,orgentity.orgentity_id
    from member,orgentity where member.type='0'
    and member.member_id=orgentity.orgentity_id"
  .
  fieldHelp=""
  .
  .
  </field>
  <field name="MARKFORDELETE"
    fieldDescription="Mark for Delete"
    type="NMTOKEN"
    sqlSelectionList=""
  >
  <datatype source="integer">
    <enumeration label="No" value="0"/>
    <enumeration label="Yes" value="1"/>
  </datatype>
  </field>
>

```

There are fields in the WebSphere Commerce database tables that use 1 and 0 for true and false. To create fields that are more intuitive to the user, enumerations of these fields can be established. Catalog Manager provides an SQL script to create an enumeration table named "NUMDESC." This script is named "createEnum.sql." An administrator can edit the sqlSelectionList attribute to make a selection list using the ENUMDESC table as shown in the example below:

```

fieldDescription="On Special"
.
.
readOnly="false"
sqlSelectionList="select description,value from
  enumdesc where columnname='ALL' and type='YESNO'"
fieldHelp=""
.
.

```

Adding field help

Field help displays at the bottom of the Web-browser window for the current field in focus. The attribute fieldHelp on the field tag in the XML form-description file or the fieldHelp key in the properties file can be used to set this value.

For example, the properties file may contain the following field-help specifications:

```

CATEGORY.MARKFORDELETE.defaultValue=No
CATEGORY.MARKFORDELETE.fieldDescription=Delete Entry

```

If no value is specified, a default message of "Enter a value for *field_name* here *field_type*" is created.













Note: If there is a properties file containing form information, it takes precedence over the entries in the XML form-description file.

Customizing search results and the work-session list

The set of columns that displays on the search-results page can be customized. There is a `showColumnInList` attribute of the `field` tag in the XML form-description file. To include a field among the search results, set this attribute to "true"; otherwise, it will not appear as one of the columns in the search-results view. The field is present when the form is displayed.

Editing the `weProcessList` file

The `weProcessList` file allows an administrator to customize the Catalog Manager utilities that are run when a Web editor work session is processed.

-  The `weProcessList.xml` file is located in the following directory:
 -  `drive:\WebSphere\CommerceServer\xml\wcwebeditor\xml`
 -  `drive:\Program Files\WebSphere\CommerceServer\xml\wcwebeditor\xml`
 -  `/usr/WebSphere/CommerceServer/xml/wcwebeditor/xml`
 -   `/opt/WebSphere/CommerceServer/xml/wcwebeditor/xml`
-  The `weProcessListOracle.xml` file is located in the following directory:
 -  `drive:\WebSphere\CommerceServer\xml\wcwebeditor\xml`
 -  `drive:\Program Files\WebSphere\CommerceServer\xml\wcwebeditor\xml`
 -  `/usr/WebSphere/CommerceServer/xml/wcwebeditor/xml`
 -  `/opt/WebSphere/CommerceServer/xml/wcwebeditor/xml`
-  The `weProcessListAS400.xml` file is located in the following directory:
 - `/instroot/xml/wcwebeditor/xml`

This file contains envelope templates for the various utilities. It can also contain references to custom applications that the administrator wishes to run.

There is a set of system variables that can be used in this file. The system variable `%-dbname%`, for example, causes the database name to be inserted in the envelope that is generated for the given invocation of a utility such as the Loader. The XML form-description file contains references to these processes that indicate which ones should be invoked for an add, edit, or delete.

Here is an example of a `weProcessList.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<processSet>
  <!-- Do not change name of extract -->
  <process name="extract"
    subsystem="com.ibm.wca.MassExtract.extract.ExtractSubSystem">
    <envelope-input xmlns='saf_params.xsd'>
      <param name="-dbname" type="scalar" value="%-dbname%"/>
      <param name="-dbuser" type="scalar" value="%-dbuser%"/>
      <param name="-dbpwd" type="scalar" value="%-dbpwd%"/>
      <param name="-outfile" type="file" reside="local" value="%tempFilePath%"/>
      <param name="-filter" type="file" reside="local" value="%tempFileURI%"/>
    </envelope-input>
  </process>
  <process name="transformer"
    subsystem="com.ibm.wca.XMLTransformer.XMLTransformerSubSystem">
    <envelope-input xmlns='saf_params.xsd'>
```

```

        <param name="-infile" type="file" reside="local" value="%tempFileURI%"/>
        <param name="-transform" type="file" reside="local"
            value="%webEditorDir%/xsl/ViewsToWCS51.XSL"/>
        <param name="-outfile" type="file" reside="local" value="%tempFilePath1%"/>
        <param name="-param" value="root=%-dbname%"/>
        <param name="-param" value="dtdname=%-dtdname%"/>
    </envelope-input>
</process>
<process name="transformerForDelete"
    subsystem="com.ibm.wca.XMLTransformer.XMLTransformerSubSystem">
    <envelope-input xmlns='saf_params.xsd'>
        <param name="-infile" type="file" reside="local" value="%tempFileURI%"/>
        <param name="-transform" type="file" reside="local"
            value="%webEditorDir%/xsl/ViewsToWCS51.XSL"/>
        <param name="-outfile" type="file" reside="local" value="%tempFilePath1%"/>
        <param name="-param" value="root=%-dbname%"/>
        <param name="-param" value="dtdname=%-dtdname%"/>
        <param name="-param" value="forDelete=true"/>
    </envelope-input>
</process>
<process name="resolver"
    subsystem="com.ibm.wca.IdResGen.IdResGenSubSystem">
    <envelope-input xmlns='saf_params.xsd'>
        <param name="-dbname" type="scalar" value="%-dbname%"/>
        <param name="-dbuser" type="scalar" value="%-dbuser%"/>
        <param name="-dbpwd" type="scalar" value="%-dbpwd%"/>
        <param name="-infile" type="file" reside="local"
            value="%previousOutFileAsURI%"/>
        <param name="-outfile" type="file" reside="local"
            value="%tempFilePath2%"/>
        <param name="-propfile" type="file" reside="local"
            value="propertyFiles.IdKeys"/>
        <param name="-method" type="scalar" value="mixed"/>
    </envelope-input>
</process>
<!-- Resolver as first process -->
<process name="resolverFirstProcess"
    subsystem="com.ibm.wca.IdResGen.IdResGenSubSystem">
    <envelope-input xmlns='saf_params.xsd'>
        <param name="-dbname" type="scalar" value="%-dbname%"/>
        <param name="-dbuser" type="scalar" value="%-dbuser%"/>
        <param name="-dbpwd" type="scalar" value="%-dbpwd%"/>
        <param name="-infile" type="file" reside="local" value="%tempFileURI%"/>
        <param name="-outfile" type="file" reside="local" value="%tempFilePath2%"/>
        <param name="-propfile" type="file" reside="local" value="propertyFiles.IdKeys"/>
        <param name="-method" type="scalar" value="mixed"/>
    </envelope-input>
</process>
<process name="loader"
    subsystem="com.ibm.wca.MassLoader.MassLoadSubSystem">
    <envelope-input xmlns='saf_params.xsd'>
        <param name="-dbname" type="scalar" value="%-dbname%"/>
        <param name="-dbuser" type="scalar" value="%-dbuser%"/>
        <param name="-dbpwd" type="scalar" value="%-dbpwd%"/>
        <param name="-commitcount" type="scalar" value="1000"/>
        <param name="-infile" type="file" reside="local" value="%previousOutFileAsURI%"/>
        <param name="-method" type="scalar" value="sqlimport"/>
        <param name="-nopriamry" type="scalar" value="insert"/>
    </envelope-input>
</process>
<process name="loaderFirstProcess"
    subsystem="com.ibm.wca.MassLoader.MassLoadSubSystem">
    <envelope-input xmlns='saf_params.xsd'>
        <param name="-dbname" type="scalar" value="%-dbname%"/>
        <param name="-dbuser" type="scalar" value="%-dbuser%"/>
        <param name="-dbpwd" type="scalar" value="%-dbpwd%"/>
        <param name="-commitcount" type="scalar" value="1000"/>
        <param name="-infile" type="file" reside="local" value="%tempFileURI%"/>
        <param name="-method" type="scalar" value="sqlimport"/>
        <param name="-nopriamry" type="scalar" value="insert"/>
    </envelope-input>
</process>
<process name="loaderForDelete"
    subsystem="com.ibm.wca.MassLoader.MassLoadSubSystem">

```

```

    <envelope-input xmlns='saf_params.xsd'>
      <param name="-dbname" type="scalar" value="%-dbname%"/>
      <param name="-dbuser" type="scalar" value="%-dbuser%"/>
      <param name="-dbpwd" type="scalar" value="%-dbpwd%"/>
      <param name="-commitcount" type="scalar" value="1000"/>
      <param name="-infile" type="file" reside="local" value="%previousOutFileAsURI%"/>
      <param name="-delete" type="scalar" value=""/>
    </envelope-input>
  </process>
  <process name="loaderForDeleteFirstProcess"
    subsystem="com.ibm.wca.MassLoader.MassLoadSubSystem">
    <envelope-input xmlns='saf_params.xsd'>
      <param name="-dbname" type="scalar" value="%-dbname%"/>
      <param name="-dbuser" type="scalar" value="%-dbuser%"/>
      <param name="-dbpwd" type="scalar" value="%-dbpwd%"/>
      <param name="-commitcount" type="scalar" value="1000"/>
      <param name="-infile" type="file" reside="local" value="%tempFileURI%"/>
      <param name="-delete" type="scalar" value=""/>
    </envelope-input>
  </process>
  <process name="saveToFile"
    cmd="cmd.exe /c c:\temp\theBatchFile.bat"
    args="-infile %tempFilePath% -dbname %-dbname%"
  />
</processSet>

```







Note: File references are case sensitive.

The following table includes a list of valid substitution variables that the application understands.

% substitution variable	Return
%-dbname%	Name of current database
%-dbuser%	Name of database user name
%-dtdname%	URI location of the DTD for the XML files
%-dbpwd%	Password for database user name
%tempFilePath%	Full path to temporary file
%tempFilePath1%	These are unique temporary file names. They may be placed in the template syntax of an envelope definition or command line. For example, if %tempFilePath% is placed in the value attribute of the -infile parameter in an envelope template, then the Web editor writes the data of the work session to the temporary file location.
%tempFilePath2%	
On SubSystem	Temporary URI
%tempFileURI%	The temporary URIs are URI to the same files represented by %tempFilePath%â%tempFilePath2%. This is not an additional set of files but a way to retrieve the same generated temporary file returned with a different syntax.
%tempFileURI1%	
%tempFileURI2%	
%previousOutFileAsURI%	Provides a URI representation of the previous tasks -outfile parameter as a URI
%webEditorDir%	Location of the Web editor installation

Editing the webeditor.xml file

The Web editor uses the webeditor.xml file as the default XSL style sheet. It is located in the following directory:

-  *drive:\WebSphere\CommerceServer\xml\wcwebeditor\xsl*
-  *drive:\Program Files\WebSphere\CommerceServer\xml\wcwebeditor\xsl*
-  */usr/WebSphere/CommerceServer/xml/wcwebeditor/xsl*
-   */opt/WebSphere/CommerceServer/xml/wcwebeditor/xsl*
-  */instroot/xml/wcwebeditor/xsl*

By editing the webeditor.xml file, an administrator can change the format of Web editor output. Here is a sample of the contents of a webeditor.xml file:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="html"/>
  <!-- Largest single line entry field value for fields larger than this a
    TEXTAREA is created -->
  <xsl:variable name="maxEntryFieldSize" select="80"/>
  <xsl:template match="/">
    <xsl:apply-templates/>
  </xsl:template>
  <!-- Read Only field -->
  <xsl:template match="readOnly" name="readOnly">
    <xsl:element name="input">
      <xsl:attribute name="name"><xsl:value-of select="@name"/></xsl:attribute>
      <xsl:attribute name="type">hidden</xsl:attribute>
      <xsl:attribute name="value"><xsl:value-of select="@defaultValue"/></xsl:attribute>
      <!--
      <xsl:attribute name="onFocus">this.blur()</xsl:attribute>
      <xsl:attribute name="style">border-style:groove</xsl:attribute>
      -->
    </xsl:element>
    <table border="1" cellpadding="0" cellspacing="0" width="155" bgcolor="#C0C0C0">
      <tr>
        <td>
          <xsl:value-of select="@defaultValue"/>
        </td>
      </tr>
    </table>
  </xsl:template>
</xsl:stylesheet>
```

Chapter 12. Working with catalogs

The following procedures describe how to work with catalogs by using the Web editor to add, modify, and delete records in the database tables containing catalog data.

Adding a record to a table using the Web editor

To add a record to a table using the Web editor, follow these steps:

1. From a Web browser, open the following URL:

```
https://host_name:8000/wcm/webeditor
```

where *host_name* is the fully qualified HTTP host name of your WebSphere Application Server.

The Web editor database logon window appears.

2. Enter your database username and password; then, click **Logon**.

The Web editor displays in the browser window with a list of table names in the left menu bar.

3. Click the appropriate hyperlink in the Add submenu of the left menu bar.

The appropriate form is displayed.

4. Type all necessary data in the form.

5. Click **Move to work session**.

The work-session results for the form are displayed. These results contain all of the edits, additions, and deletions that you have made but not cleared or processed during this Web editor work session.

6. If you want to remove any record changes from the work session, check the box in front of each record change that you want to remove and click **Clear selected**.

7. Click **Process work session** to submit the selected changes to the database.

A status page displays a message stating that the process was successful.

8. Navigate to the Web site, click the appropriate hyperlinks, and verify that your changes have been made.

Modifying a record in a table using the Web editor

To modify a record in a table using the Web editor, follow these steps:

1. From a Web browser, open the following URL:

```
https://host_name:8000/wcm/webeditor
```

where *host_name* is the fully qualified HTTP host name of your WebSphere Application Server.

The Web editor database logon window appears.

2. Enter your database username and password; then, click **Logon**.
The Web editor displays in the browser window with a list of table names in the left menu bar.
3. Click the appropriate hyperlink in the Search submenu of the left menu bar.
The appropriate search page displays.
4. Specify your search criteria by doing the following:
 - a. Select the check box beside each attribute that you want to specify in your search.
 - b. Use the appropriate drop-down menu for each selected attribute to select the logic that you want to use in the search.
 - c. In the next field for each selected attribute, type or select the value that you want to use in the search.
5. Click **Find**.
This submits the search criteria to the Web editor.
A status page displays with the number of records meeting your search criteria.
6. Do one of the following:
 - Click **Load data** to see a list of the records found.
The Web editor displays a list of the records retrieved from your query.
Go to Step 7.
 - Click **New search** to return to the search page.
Go back to Step 4.
7. Select the record that you want to edit.
The appropriate form is displayed.
8. Scroll down to the field that you want to edit, and change its content.
9. Click **Move to work session**.
The work-session results for the form are displayed. These results contain all of the edits, additions, and deletions that you have made but not cleared or processed during this Web editor work session.
10. If you want to remove any record changes from the work session, check the box in front of each record change that you want to remove and click **Clear selected**.
11. Click **Process work session** to submit the selected changes to the database.
A status page displays a message stating that the process was successful.
12. Navigate to the Web site, click the appropriate hyperlinks, and verify that your changes have been made.

Deleting a record from a table using the Web editor

To delete a record from a table using the Web editor, follow these steps:

1. From a Web browser, open the following URL:

```
https://host_name:8000/wcm/webeditor
```

where *host_name* is the fully qualified HTTP host name of your WebSphere Application Server.

The Web editor database logon window appears.

2. Enter your database username and password; then, click **Logon**.

The Web editor displays in the browser window with a list of table names in the left menu bar.

3. Click the appropriate hyperlink in the Search submenu of the left menu bar. The appropriate search page is displayed.

4. Specify your search criteria by doing the following:

- a. Select the check box beside each attribute that you want to specify in your search.
- b. Use the appropriate drop-down menu for each selected attribute to select the logic that you want to use in the search.
- c. In the next field for each selected attribute, type or select the value that you want to use in the search.

5. Click **Find**.

This submits the search criteria to the Web editor.

A status page displays with the number of records meeting your search criteria.

6. Do one of the following:

- Click **Load data** to see a list of the records found.

The Web editor displays a list of the records retrieved from your query.

Go to Step 7.

- Click **New search** to return to the search page.

Go back to Step 4.

7. Select the check box beside each record that you want to delete.

8. Click **Move to delete list**.

9. Click the appropriate hyperlink in the Work Session submenu of the left menu bar.

The work-session results for the form are displayed. These results contain all of the edits, additions, and deletions that you have made but not cleared or processed during this Web editor work session.

10. If you want to remove any record changes from the work session, check the box in front of each record change that you want to remove and click **Clear selected**.

11. Click **Process work session** to submit the changes to the database.

A status page displays a message stating that the process was successful.

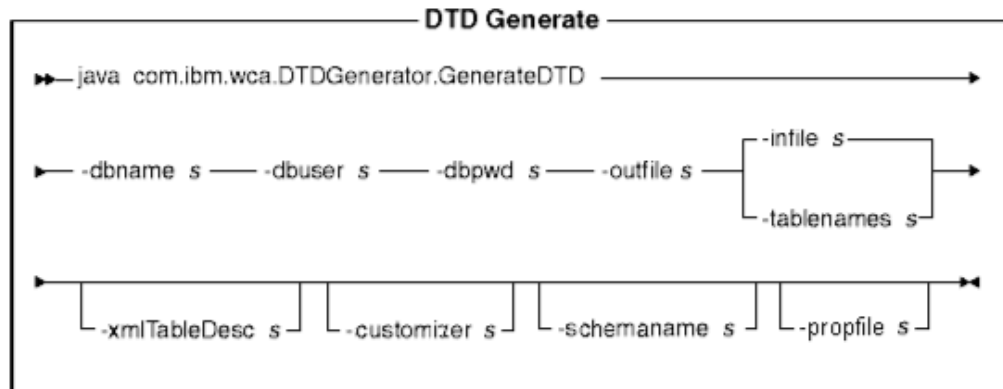
12. Navigate to the Web site, click the appropriate hyperlinks, and verify that your changes have been made.

Part 4. Command Reference

Chapter 13. DTD Generate command

This command creates DTD and schema files for use with the Loader package.

DTD Generate command for Windows, AIX, Linux, and Solaris systems



Notes:

1. The above diagram is intended primarily as a reference for the command parameters. The command file or script provided for this command and listed under Chapter 10, "Configuring Loader package commands and scripts" on page 71 acts as a wrapper to the actual Java command and accepts the same parameters; therefore, it is recommended that you use the command file or script rather than invoke the Java command directly.
2. Filenames specified as parameters for this command can be preceded by relative or absolute paths.

Parameter values:

-dbname

Name of the target database

-dbuser

Name of the user connecting to the database

-dbpwd

Password for the user connecting to the database

-outfile

Name of the output DTD file

-infile Name of an input file containing a database-table name on each line

-tablenames

Names of tables, separated by commas

-xmlTableDesc

File path of the schema file to be created

-customizer

Name of the customizer property file to be used. DB2ConnectionCustomizer.properties is the default file. The customizer property file can be specified as shown in the following example:

```
-customizer d:\WebSphere\CommerceServer\prop\dttdgen.properties
```

If this file exists in a directory specified in the classpath system-environment variable, the same file can be specified as shown in the following example:

```
-customizer dttdgen
```

-schemaname

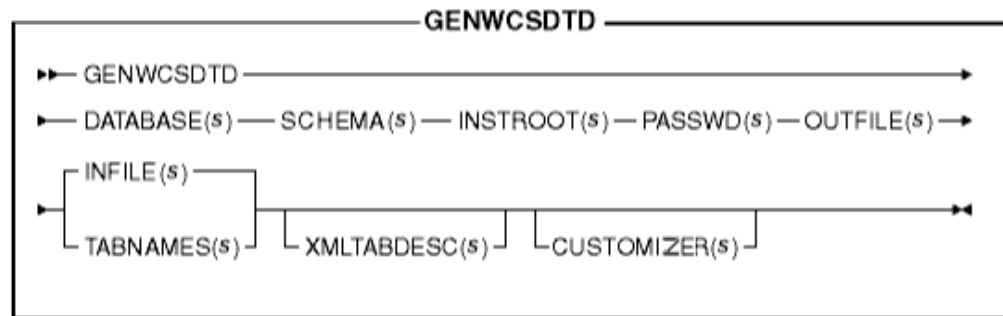
Name of the target database schema. This parameter is optional.

If this parameter is not specified when running the command, the command looks for a name=value pair in the customizer property file that specifies the value of SchemaName. If this pair is present in the property file, the command uses the value specified. If neither a command-line nor a property-file specification for this parameter exists, the command defaults to the name of the database user.

-propfile

File containing properties such as an external properties file where help text, default values, and field-description information can be stored for a Web editor form description

DTD Generate command for iSeries systems



Note: Filenames specified as parameters for this command can be preceded by relative or absolute paths.

Parameter values:

DATABASE

Name of the target database, as displayed in the relational database directory

SCHEMA

Name of the target database schema; this is the same as the instance name

INSTROOT

Full name of the WebSphere Commerce instance root path, such as
/QIBM/UserData/WebCommerce/instances/instance_name

PASSWD

Password for the WebSphere Commerce instance

OUTFILE

Name of the output DTD file

INFILE

Name of an input file containing a database-table name on each line

TABNAMES

Names of tables, separated by commas

XMLTABDESC

File path of the schema file to be created. This parameter is optional.

CUSTOMIZER

Name of the customizer property file to be used. The default file is *ISeries_GENWCSDTD_Customizer.properties*. The customizer property file can be specified as shown in the following example:

```
CUSTOMIZER(/wc/prop/dtdgen.properties)
```

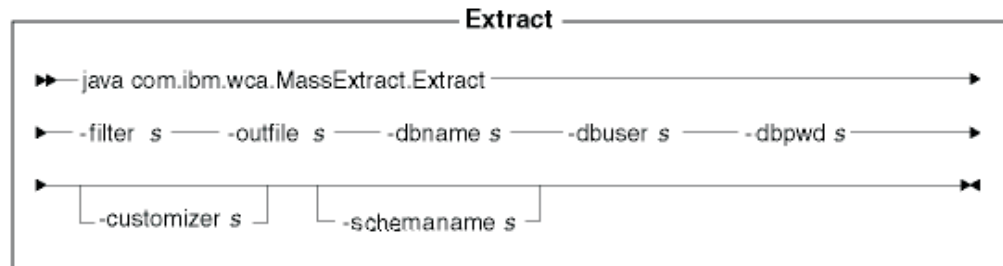
If this file exists in a directory specified in the classpath system-environment variable, the same file can be specified as shown in the following example:

```
CUSTOMIZER(dtdgen)
```

Chapter 14. Extract command

This command extracts a selected subset of data from a database in the form of an XML file.

Extract command for Windows, AIX, Linux, and Solaris systems



Notes:

1. The above diagram is intended primarily as a reference for the command parameters. The command file or script provided for this command and listed under Chapter 10, “Configuring Loader package commands and scripts” on page 71 acts as a wrapper to the actual Java command and accepts the same parameters; therefore, it is recommended that you use the command file or script rather than invoke the Java command directly.
2. Filenames specified as parameters for this command can be preceded by relative or absolute paths.

Parameter values:

-filter Name of the extraction-filter file

-outfile

Name for the output XML file where the extracted data will be stored

-dbname

Name of the database from which data is being extracted

-dbuser

Database user name for the database from which data is being extracted

-dbpwd

Password associated with the user name for the database from which data is being extracted

-customizer

Name of the customizer property file to be used.

DB2ConnectionCustomizer.properties is the default file. The customizer property file can be specified as shown in the following example:

```
-customizer d:\WebSphere\CommerceServer\prop\extract.properties
```

If this file exists in a directory specified in the classpath system-environment variable, the same file can be specified as shown in the following example:

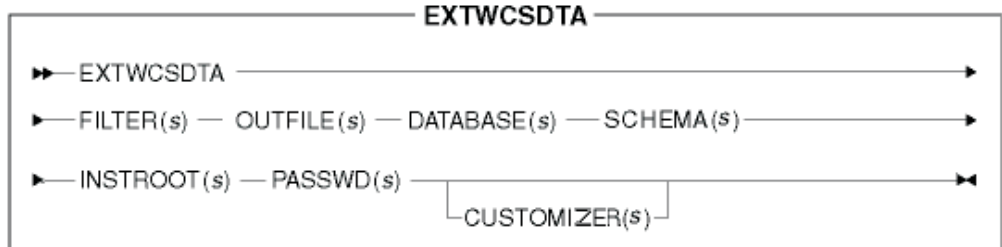
```
-customizer extract
```

-schemaname

Name of the database schema from which data is being extracted. This parameter is optional.

If this parameter is not specified when running the command, the command looks for a name=value pair in the customizer property file that specifies the value of SchemaName. If this pair is present in the property file, the command uses the value specified. If neither a command-line nor a property-file specification for this parameter exists, the command defaults to the schema name of the table in the database.

Extract command for iSeries systems



Note: Filenames specified as parameters for this command can be preceded by relative or absolute paths.

Parameter values:

FILTER

Name of the extraction-filter file

OUTFILE

Name for the output XML file where the extracted data will be stored

DATABASE

Name of the database from which data is being extracted as displayed in the relational database directory

SCHEMA

Name of the database schema from which data is being extracted; this is the same as the instance name

INSTROOT

Full name of the WebSphere Commerce instance root path, such as /QIBM/UserData/WebCommerce/instances/*instance_name*

PASSWD

Password for the WebSphere Commerce instance

CUSTOMIZER

Name of the customizer property file to be used. The default file is ISeries_EXTWCSDTA_Customizer.properties. The customizer property file can be specified as shown in the following example:

CUSTOMIZER(/wc/prop/extract.properties)

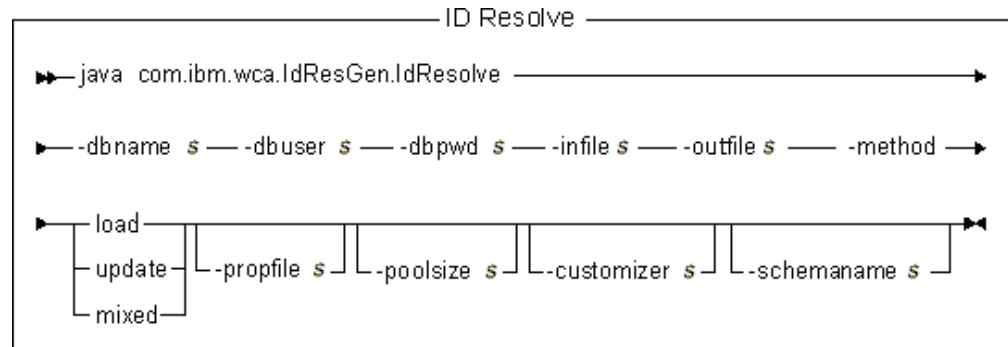
If this file exists in a directory specified in the classpath system-environment variable, the same file can be specified as shown in the following example:

CUSTOMIZER(extract)

Chapter 15. ID Resolve command

This command generates identifiers for XML elements that require them prior to loading into a database.

ID Resolve command for Windows, AIX, Linux, and Solaris systems



Notes:

1. The above diagram is intended primarily as a reference for the command parameters. The command file or script provided for this command and listed under Chapter 10, "Configuring Loader package commands and scripts" on page 71 acts as a wrapper to the actual Java command and accepts the same parameters; therefore, it is recommended that you use the command file or script rather than invoke the Java command directly.
2. Filenames specified as parameters for this command can be preceded by relative or absolute paths.

Parameter values:

-dbname

Name of the target database

-dbuser

Name of the user connecting to the database

-dbpwd

Password for the user connecting to the database

-infile Name of the input XML document containing table records

-outfile

Name of the output XML file to be produced; this file can be used as input to the Loader

-method

Method to be used in processing the input file

- Use the load method to process the input file if *all* records in the file *do not exist* in the database.
- Use the update method to process the input file if *all* records in the file *exist* in the database.
- Use the mixed method to process the input file if *only some* records in the file *exist* in the database.

The default method is load.

-propfile

Text file containing Java properties in the form of name=value pairs. This property file sets the way in which the ID Resolver resolves identifiers. It is used to describe which columns of a primary entry should be used as lookups for tables that require the identifier of a primary row. This file defines the column names for foreign-key identifier lookup and the select predicate for main table (such as CATEGORY and PRODUCT) queries. You can omit entries in this file for tables that have a defined unique index that does not include the identifier. This parameter is optional.

IdResolveKeys.properties is the default file. This property file can be specified as shown in either of the following examples:

```
-propfile d:\WebSphere\CommerceServer\prop\idresprop.properties  
-propfile d:\WebSphere\CommerceServer\prop\idresprop
```

If this file exists in the current directory, the same file can be specified as shown in the following example:

```
-propfile idresprop.properties
```

If this file exists in a directory specified in the classpath system-environment variable, the same file can be specified as shown in the following example:

```
-propfile idresprop
```

-poolsize

Number of identifiers to be reserved. The default number is 50.

-customizer

Name of the customizer property file to be used.

DB2ConnectionCustomizer.properties is the default file. The customizer property file can be specified as shown in either of the following examples:

```
-customizer d:\WebSphere\CommerceServer\prop\idres.properties  
-customizer d:\WebSphere\CommerceServer\prop\idres
```

If this file exists in the current directory, the same file can be specified as shown in the following example:

```
-customizer idres.properties
```

If this file exists in a directory specified in the classpath system-environment variable, the same file can be specified as shown in the following example:

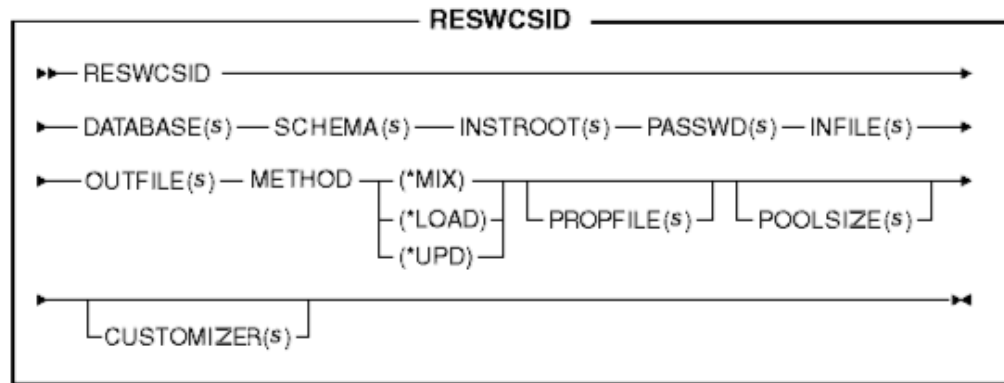
```
-customizer idres
```

-schemaname

Name of the target database schema. This parameter is optional.

If this parameter is not specified when running the command, the command looks for a name=value pair in the customizer property file that specifies the value of SchemaName. If this pair is present in the property file, the command uses the value specified. If neither a command-line nor a property-file specification for this parameter exists, the command defaults to the schema name of the table in the database.

ID Resolve command for iSeries systems



Note: Filenames specified as parameters for this command can be preceded by relative or absolute paths.

Parameter values:

DATABASE

Name of the target database, as displayed in the relational database directory

SCHEMA

Name of the target database schema; this is the same as the instance name

INSTROOT

Full name of the WebSphere Commerce instance root path, such as
/QIBM/UserData/WebCommerce/instances/*instance_name*

PASSWD

Password for the WebSphere Commerce instance

INFILE

Name of the input XML document containing table records

OUTFILE

Name of the output XML file to be produced; this file can be used as input to the Loader

METHOD

Method to be used in processing the input file

- Use the load method (*LOAD) to process the input file if *all* records in the file *do not exist* in the database.
- Use the update method (*UPD) to process the input file if *all* records in the file *exist* in the database.
- Use the mixed method (*MIX) to process the input file if *only some* records in the file *exist* in the database.

PROPFIELD

Text file containing Java properties in the form of name=value pairs. This property file sets the way in which the ID Resolver resolves identifiers. It is used to describe which columns of a primary entry should be used as lookups for tables that require the identifier of a primary row. This file defines the column names for foreign-key identifier lookup and the select predicate for main table (such as CATEGORY and PRODUCT) queries. You can omit entries in this file for tables that have a defined unique index that does not include the identifier. This parameter is optional.

IdResolveKeys.properties is the default file. This property file can be specified as shown in either of the following examples:

```
PROPFIELD(/wc/prop/idresprop.properties)
```

```
PROPFIELD(/wc/prop/idresprop)
```

If this file exists in the current directory, the same file can be specified as shown in the following example:

```
PROPFIELD(idresprop.properties)
```

If this file exists in a directory specified in the classpath system-environment variable, the same file can be specified as shown in the following example:

```
PROPFIELD(idresprop)
```

POOLSIZE

Number of identifiers to be reserved. The default number is 50.

CUSTOMIZER

Name of the customizer property file to be used. The default file is ISeries_RESWCSID_Customizer.properties. The customizer property file can be specified as shown in either of the following examples:

```
CUSTOMIZER(/wc/prop/idres.properties)
```

```
CUSTOMIZER(/wc/prop/idres)
```

If this file exists in the current directory, the same file can be specified as shown in the following example:

```
CUSTOMIZER(idres.properties)
```

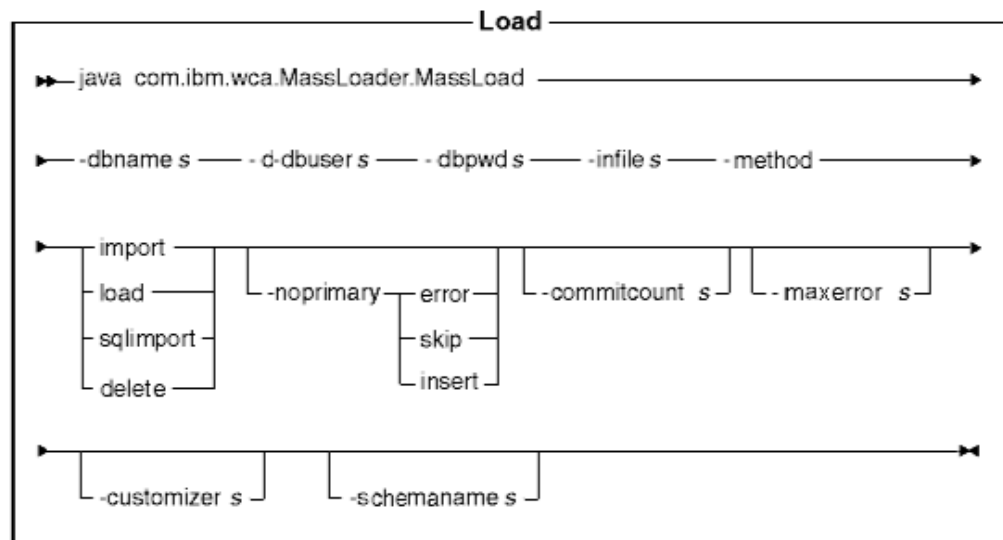
If this file exists in a directory specified in the classpath system-environment variable, the same file can be specified as shown in the following example:

```
CUSTOMIZER(idres)
```

Chapter 16. Load command

This command loads an XML input file into a target database.

Load command for Windows, AIX, Linux, and Solaris systems



Notes:

1. The above diagram is intended primarily as a reference for the command parameters. The command file or script provided for this command and listed under Chapter 10, "Configuring Loader package commands and scripts" on page 71 acts as a wrapper to the actual Java command and accepts the same parameters; therefore, it is recommended that you use the command file or script rather than invoke the Java command directly.
2. Filenames specified as parameters for this command can be preceded by relative or absolute paths.

Parameter values:

-dbname

Name of the target database

-dbuser

Name of the user connecting to the database

-dbpwd

Password for the user connecting to the database

-infile Name of the input XML file

-method

Mode of operation for the Loader to use when inserting data into the database

- The load method uses the native loader from the database vendor. You can use the load method for both local and remote Oracle databases; but the load method can only be used for local DB2 databases.
- Although the import method can be used to load data into local or remote databases, it is usually used to load data into remote DB2 databases. This method uses the import or update option if it is available from the database vendor. If you specify this method for a database in which the import or update option is not available, such as Oracle, SQL statements using JDBC are used to update the database.
- The SQL import (sqlimport) method can be used with both local and remote databases.
- The delete method deletes data from the database.

If you are using Product Advisor search-space synchronization, you must use either the sqlimport or the delete method.

-noprimary

Action the Loader must take when the primary key is missing for a record in the input file. The error option indicates that it should report the missing primary key as an error and terminate. The skip option skips any record in the input file that does not have a primary key. The insert option tries to process (insert or delete) the data. The default action is error.

-commitcount

Number of records processed before the database commit occurs when using the SQL update method of operation. The default number is 1.

-maxerror

Number of errors after which the Loader will terminate in the SQL update method of operation

-customizer

Name of the customizer property file to be used.

MassLoadCustomizer.properties is the default file. The customizer property file can be specified as shown in the following example:

```
-customizer d:\WebSphere\CommerceServer\prop\ml.properties
```

If this file exists in a directory specified in the classpath system-environment variable, the same file can be specified as shown in the following example:

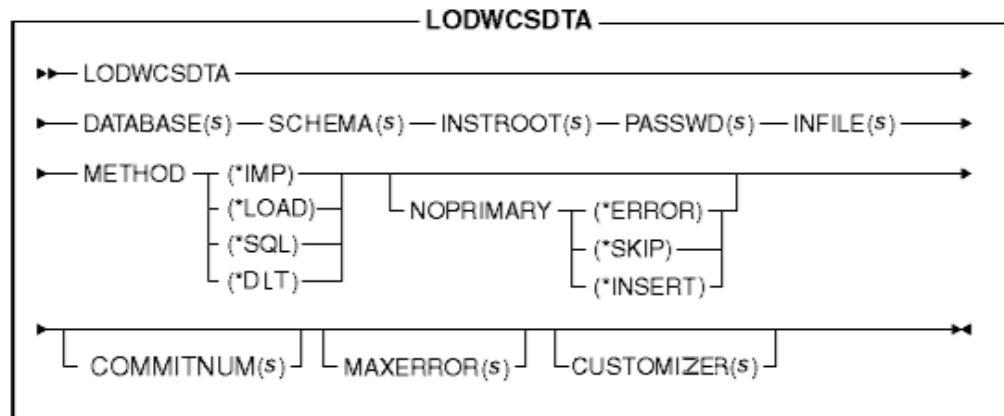
```
-customizer ml
```

-schemaname

Name of the target database schema. This parameter is optional.

If this parameter is not specified when running the command, the command looks for a name=value pair in the customizer property file that specifies the value of SchemaName. If this pair is present in the property file, the command uses the value specified. If neither a command-line nor a property-file specification for this parameter exists, the command defaults to the schema name of the table in the database.

Load command for iSeries systems



Note: Filenames specified as parameters for this command can be preceded by relative or absolute paths.

Parameter values:

DATABASE

Name of the target database as displayed in the relational database directory

SCHEMA

Name of the target database schema; this is the same as the instance name

INSTROOT

Full name of the WebSphere Commerce instance root path, such as
/QIBM/UserData/WebCommerce/instances/instance_name

PASSWD

Password for the WebSphere Commerce instance

INFILE

Name of the input XML file

METHOD

Mode of operation for the Loader to use when inserting data into the database

- The load method (*LOAD) uses the native loader from the database vendor. You can use the load method (*LOAD) for both local and remote Oracle databases; but the load method (*LOAD) can only be used for local DB2 databases.
- Although the import (*IMP) method can be used to load data into local or remote databases, it is usually used to load data into remote DB2 databases. This method uses the import or update option if it is available from the database vendor. If the import or update option is not available, SQL statements using JDBC are used to update the database.
- The SQL import (*SQL) method can be used with both local and remote databases.
- The delete (*DLT) method deletes data from the database.

NOPRIMARY

Action that the Loader must take when the primary key is missing for a record in the input file. The error option (*ERROR) indicates that it should report the missing primary key as an error and terminate. The skip option (*SKIP) skips any record in the input file that does not have a primary key. The insert option (*INSERT) tries to process (insert or delete) the data. The default action is error.

COMMITNUM

Number of records processed before the database commit occurs when using the SQL update method of operation. The default number is 1.

MAXERROR

Number of errors after which the Loader will terminate in the SQL update method of operation

CUSTOMIZER

Name of the customizer property file to be used. The default file is `ISeries_LODWCSDTA_Customizer.properties`. The customizer property file can be specified as shown in the following example:

```
CUSTOMIZER(/wc/prop/m1.properties)
```

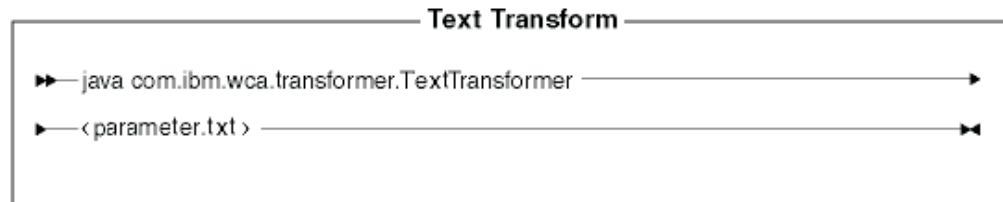
If this file exists in a directory specified in the classpath system-environment variable, the same file can be specified as shown in the following example:

```
CUSTOMIZER(m1)
```

Chapter 17. Text Transform command

This command transforms data between a character-delimited variable format and an XML format.

Text Transform command for Windows, AIX, Linux, and Solaris systems



Note: The above diagram is intended primarily as a reference for the command parameters. The command file or script provided for this command and listed under Chapter 10, "Configuring Loader package commands and scripts" on page 71 acts as a wrapper to the actual Java command and accepts the same parameters; therefore, it is recommended that you use the command file or script rather than invoke the Java command directly.

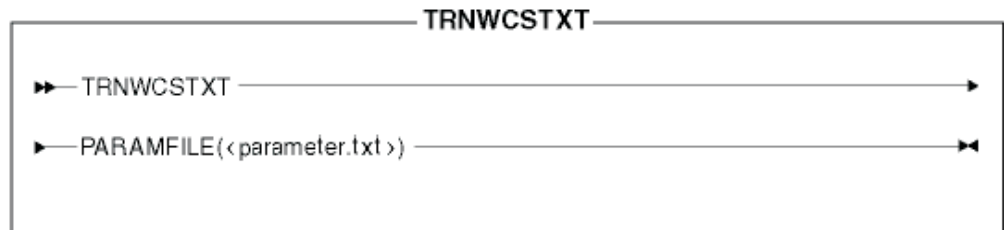
Parameter values

The following values are specified and separated by commas in a parameter file (*parameter.txt*):

- input file
Name of the file to be transformed
- schema file
Name of the XML schema file to be used in the transformation
- output file
Name for the output file in which the transformed data will be stored
- transformation method
Method to be used in adding the data to the output file. Specify **Create** if a new file is to be created; or specify **Append** if the output data is to be appended to an existing data file.

Note: This file is also referred to as a "manifest" or "command" file.

Text Transform command for iSeries systems



Parameter values

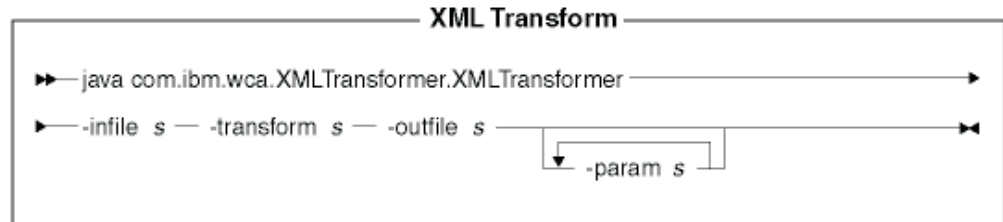
The following values are specified and separated by commas in a parameter file (*parameter.txt*):

- input file
Name of the file to be transformed
- schema file
Name of the XML schema file to be used in the transformation
- output file
Name for the output file in which the transformed data will be stored
- transformation method
Method to be used in adding the data to the output file. Specify **Create** if a new file is to be created; or specify **Append** if the output data is to be appended to an existing data file.

Chapter 18. XML Transform command

This command converts an XML file into an alternate XML format.

XML Transform command for Windows, AIX, Linux, and Solaris systems



Notes:

1. The above diagram is intended primarily as a reference for the command parameters. The command file or script provided for this command and listed under Chapter 10, "Configuring Loader package commands and scripts" on page 71 acts as a wrapper to the actual Java command and accepts the same parameters; therefore, it is recommended that you use the command file or script rather than invoke the Java command directly.
2. Filenames specified as parameters for this command can be preceded by relative or absolute paths.

Parameter values:

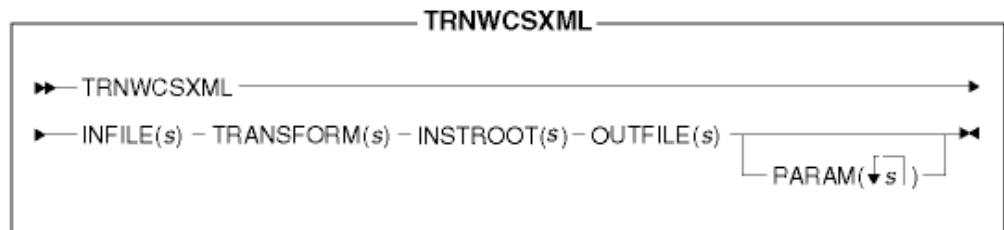
-infile Name of the file to be transformed

-transform
Name of the transform XSL rule file

-outfile
Name for the output XML file in which the transformed data will be stored

-param
Parameter to be passed to the XSL rule file. *This parameter is optional.*
This parameter can be specified multiple times to pass multiple "name=value" pairs.

XML Transform command for iSeries systems



Note: Filenames specified as parameters for this command can be preceded by relative or absolute paths.

Parameter values:

INFILE

Name of the file to be transformed

TRANSFORM

Name of the transform XSL rule file

INSTROOT

Full name of the WebSphere Commerce instance root path, such as
`/QIBM/UserData/WebCommerce/instances/instance_name`

OUTFILE

Name for the output XML file in which the transformed data will be stored

PARAM

Parameter to be passed to the XSL rule file. *This parameter is optional.*
The string can contain multiple values to pass multiple "name=value" pairs.

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 Canada Ltd.
Office of the Lab Director
8200 Warden Avenue, Markham, Ontario L6G 1C7
Canada

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 measurement 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.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

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. 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 IBM's application programming interfaces.

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

©Copyright International Business Machines Corporation 2001.
Portions of this code are derived from IBM Corp. Sample Programs.
©Copyright IBM Corp. 2000, 2001. All rights reserved.

If you are viewing this information in softcopy, the photographs and color illustrations may not appear.

Trademarks and service marks

The following terms are trademarks of International Business Machines Corporation in the United States or other countries or both:

AIX
DB2
DB2 Universal Database
IBM
iSeries
OS/400
WebSphere

Microsoft, Windows, Windows NT, and Windows 2000 are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Oracle is a registered trademark of Oracle Corporation in the United States, other countries, or both.

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

Other company, product, and service names may be trademarks or service marks of others.



Printed in U.S.A.