

Oracle Utilities Energy Information Platform

Configuration Guide

Release 1.6.1.17 for Windows

E18200-18

February 2014

Oracle Utilities Energy Information Platform/Energy Information Platform Configuration Guide, Volume 1,
Release 1.6.1.17 for Windows

E18200-18

Copyright © 1999, 2012 Oracle and/or its affiliates. All rights reserved.

Primary Author: Lou Prosperi

Contributor: Steve Pratt

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

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

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

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are “commercial computer software” pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

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

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

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

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

NOTIFICATION OF THIRD-PARTY LICENSES

Oracle Utilities software contains third party, open source components as identified below. Third- party license terms and other third-party required notices are provided below.

License: Apache 1.1

Module: Crimson v1.1.1, Xalan J2

Copyright © 1999-2000 The Apache Software Foundation. All rights reserved.

Use of Crimson 1.1.1 and Xalan J2 within the product is governed by the following (Apache 1.1):

(1) Redistributions of source code must retain the above copyright notice, this list of conditions and the disclaimer below. (2) Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the disclaimer below in the documentation and/or other materials provided with the distribution. (3) The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>). " Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear. (4) Neither the component name nor Apache Software Foundation may be used to endorse or promote products derived from the software without specific prior written permission. (5) Products derived from the software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

License: CoolServlets.com

Module: CS CodeViewer v1.0 (Sun JRE Component)

Copyright © 1999 by CoolServlets.com

Use of this module within the product is governed by the following:

(1) Redistributions of source code must retain the above copyright notice, this list of conditions and the disclaimer below. (2) Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the disclaimer below in the documentation and/or other materials provided with the distribution. (3) Neither the component name nor CoolServlets.com may be used to endorse or promote products derived from the software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY COOLSERVLTS.COM AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE."

License: Justin Frankel, justin@nullsoft.com

Module: NSIS 1.0j (Sun JRE Component)

Use of this module within the product is governed by the following:

(1) The origin of the module must not be misrepresented, and Oracle may not claim that it wrote the original software. If Oracle uses this module in a product, an acknowledgment in the product documentation is appreciated but not required. (2) Altered source versions of the module must be plainly marked as such, and

must not be misrepresented as being the original software. (3) The following notice may not be removed or altered from any source distribution: "Justin Frankel justin@nullsoft.com".

License: ICU4j License

Module: ICU4j

Copyright © 1995-2001 International Business Machines Corporation and others. All rights reserved.

Oracle may use the software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the software, and to permit persons to whom the software is furnished to do so, provided that the above copyright notice and the permission notice appear in all copies of the software and that both the above copyright notice and the permission notice appear in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

License: Info-ZIP

Module: INFO-ZIP ZIP32.DLL (Binary Form)

Copyright (c) 1990-2005 Info-ZIP. All rights reserved

Use of this dll within the product is governed by the following:

(1) Redistributions of source code must retain the above copyright notice, this list of conditions and the definition and disclaimer below. (2) Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the definition and disclaimer below in the documentation and/or other materials provided with the distribution. The sole exception to this condition is redistribution of a standard UnZipSFX binary (including SFXWiz) as part of a self-extracting archive; that is permitted without inclusion of this license, as long as the normal SFX banner has not been removed from the binary or disabled. (3) Altered versions—including, but not limited to, ports to new operating systems, existing ports with new graphical interfaces, and dynamic, shared, or static library versions—must be plainly marked as such and must not be misrepresented as being the original source. Such altered versions also must not be misrepresented as being Info-ZIP releases—including, but not limited to, labeling of the altered versions with the names "Info-ZIP" (or any variation thereof, including, but not limited to, different capitalizations), "Pocket UnZip," "WiZ" or "MacZip" without the explicit permission of Info-ZIP. Such altered versions are further prohibited from misrepresentative use of the Zip-Bugs or Info-ZIP e-mail addresses or of the Info-ZIP URL(s). (4) Info-ZIP retains the right to use the names "Info-ZIP," "Zip," "UnZip," "UnZipSFX," "WiZ," "Pocket UnZip," "Pocket Zip," and "MacZip" for its own source and binary releases.

[Definition]: For the purposes of this copyright and license, "Info-ZIP" is defined as the following set of individuals:

Mark Adler, John Bush, Karl Davis, Harald Denker, Jean-Michel Dubois, Jean-loup Gailly, Hunter Goatley, Ed Gordon, Ian Gorman, Chris Herborth, Dirk Haase, Greg Hartwig, Robert Heath, Jonathan Hudson, Paul Kienitz, David Kirschbaum, Johnny Lee, Onno van der Linden, Igor Mandrichenko, Steve P. Miller, Sergio Monesi, Keith Owens, George Petrov, Greg Roelofs, Kai Uwe Rommel, Steve Salisbury, Dave Smith, Steven M. Schweda, Christian Spieler, Cosmin Truta, Antoine Verheijen, Paul von Behren, Rich Wales, Mike White

[Disclaimer]: "This software is provided "as is," without warranty of any kind, express or implied. In no event shall Info-ZIP or its contributors be held liable for any direct, indirect, incidental, special or consequential damages arising out of the use of or inability to use this software."

License: Paul Johnston

Modules: md5.js

Copyright (C) Paul Johnston 1999 - 2002

Use of these modules within the product is governed by the following:

(1) Redistributions of source code must retain the above copyright notice, this list of conditions and the disclaimer below. (2) Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the disclaimer below in the documentation and/or other materials provided with the distribution. (3) Neither the component name nor the names of the copyright holders and contributors may be used to endorse or promote products derived from the software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

License: Jef Poskanzer

Modules: DES, 3xDES (Sun JRE Components)

Copyright © 2000 by Jef Poskanzer <jef@acme.com>. All rights reserved

Use of these modules within the product is governed by the following:

(1) Redistributions of source code must retain the above copyright notice, this list of conditions and the disclaimer below. (2) Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the disclaimer below in the documentation and/or other materials provided with the distribution. (3) Neither the component name nor the name of Jef Poskanzer may be used to endorse or promote products derived from the software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

License: Sun Microsystems, Inc.

Modules: Sun Swing Tutorials

Copyright© 1995-2006 Sun Microsystems, Inc. All Rights Reserved.

Use of these modules within the product is governed by the following:

(1) Redistributions of source code must retain the above copyright notice, this list of conditions and the disclaimer below. (2) Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the disclaimer below in the documentation and/or other materials provided with the distribution. (3) Neither the component name nor the name of Sun Microsystems, Inc. and contributors may be used to endorse or promote products derived from the software without specific prior written permission. (4) Oracle must acknowledge that the software is not designed, licensed or intended for use in the design, construction, operation or maintenance of any nuclear facility.

THIS SOFTWARE IS PROVIDED "AS IS," WITHOUT A WARRANTY OF ANY KIND. ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SUN MICROSYSTEMS, INC. ("SUN") AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR

PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE THIS SOFTWARE, EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

License: Tom Wu

Module: jsbn library

Copyright © 2003-2005 Tom Wu. All rights reserved

Use of this module within the product is governed by the following:

(1) Redistributions of source code must retain the above copyright notice, this list of conditions and the disclaimer below. (2) Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the disclaimer below in the documentation and/or other materials provided with the distribution.

THE SOFTWARE IS PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL TOM WU BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY THEORY OF LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Contents

What's New

New Features in the Oracle Utilities Energy Information Platform Configuration Guide	1-i
New Features for Release 1.6.0.0	1-i
New Features for Release 1.6.1.0	1-ii
New Features for Release 1.6.1.17	1-ii

Chapter 1

Overview	1-1
Configuration Overview	1-2
What is this book?	1-3
International and Multiple Currency Support	1-5
Locales and Languages	1-5
Date and Numeric Presentation	1-6
Currency Presentation	1-6
Multiple Currency Support	1-8
File Formats	1-9
Database Support	1-9
Notes, Cautions, and Warnings	1-10

Part One

Configuration	1-i
---------------------	-----

Chapter 2

Configuration Files	2-1
LODESTAR.CFG	2-2
Allowable Settings	2-3
EXTENDEDSTATUSES.CFG.XML Configuration File	2-12
IDV.CFG.XML	2-15
INTDCONFIG.CFG.XML	2-17
LOCALES.CFG.XML	2-22
LSCALENDAR.CFG.XML	2-23
LSDB.CFG.XML	2-26
LSLOGGER.CFG.XML	2-27
LSRELAY.CFG.XML	2-34
LSREPORTMONITOR.CFG.XML	2-35
LSSCHDLR.CFG.XML	2-41
LSSECURE.CFG.XML	2-42
LTMH.CFG.XML	2-46
Other Configuration Files	2-52
Tools Files (*.TLS)	2-52
Encrypting Configuration Files	2-54

Chapter 3

Setting Up Energy Information Platform Database Records	3-1
Setting Up Energy Information Platform Lookup Tables.....	3-2
Account Statuses	3-2
Bill Determinants	3-2
End Use	3-3
Industrial Classifications	3-3
Jurisdiction	3-3
LSCurrency	3-3
Operating Company	3-4
Revenue Code.....	3-4
Second Per Interval.....	3-4
Standard Industrial Classification (SIC) Code	3-5
Unit of Measure (UOM)	3-5
Overriding and/or Extending the Default UOM List	3-6
Setting Up Other Records.....	3-7
Address	3-8
Address Format.....	3-8
Business Calendar	3-9
Business Day.....	3-9
CIS Account.....	3-9
Contact Method Type	3-9
Contact Purpose.....	3-10
Directory.....	3-10
Factor	3-11
Factor Values	3-11
Group Category.....	3-12
Interrogations	3-12
Interval Data Sources	3-12
LS Address	3-13
LS Market	3-13
LS Premise	3-14
Market Attribute.....	3-14
Market Attribute Type.....	3-14
Market Participant Relationship Type	3-14
Market Participant Type.....	3-15
MV90 Status Code Map.....	3-15
Organization	3-15
Override.....	3-15
Participant Type	3-16
Web Service	3-16
Web Service Type	3-16

Chapter 4

Configuring Interval Data Viewer and Interval Data Manager	4-1
Configuring Interval Data Viewer	4-2
Prepare Interval Data	4-2
Set Up Interval Data Sources.....	4-2
Assign Data and Feature Privileges.....	4-2
Configuring Interval Data Manager.....	4-3
Prepare Interval Data	4-3
Set Up Interval Data Sources.....	4-4
Assign Data and Feature Privileges.....	4-4
Set Up Validation Environment File (Optional).....	4-4

Chapter 5

Configuring Work Queues.....	5-1
Work Queues Tables Overview	5-2
Work Queues Actions	5-3
Opening a New Item.....	5-3
Listing Items	5-3
Assigning an Item	5-4
Unassigning an Item	5-4
Updating an Item	5-5
Resolving an Item	5-5
Approving an Item	5-5
Rejecting an Item	5-6
Closing an Item	5-7
Reopening an Item.....	5-7
Defining Your Work Queues Configuration	5-8
Defining Work Queues	5-8
Defining Work Queue Types	5-8
Configuring Work Queues and Work Queue Types	5-11
Setting Up Lookup Tables.....	5-11
Setting Up Work Queue Types.....	5-12
Creating Work Queue Items.....	5-20
Using Oracle Utilities Billing Component	5-20
Using the Oracle Utilities Rules Language.....	5-22
Using External Applications.....	5-24

Chapter 6

Setting Up, Configuring, and Running the Energy Information Platform Adapter.....	6-1
Defining the Adapter Implementation.....	6-2
Defining Business Requirements.....	6-2
Defining and Naming Adapter Servers	6-6
Installing the Adapter Software.....	6-7
Install the Adapter Software.....	6-7
Set Up the LTMH.CFG.XML File.....	6-7
Install JMS Servers and Queues (Optional).....	6-8
Configuring the Adapter	6-9
Configuring Adapter Components.....	6-9
Configuring Business Rules.....	6-11
Configuring Adapter Services	6-17
Using Adapter Data Converters.....	6-22
CSV Parsers	6-22
XML Converters	6-31
Interval Data Converters	6-32
Payload Splitter.....	6-34
ZIP Converters.....	6-38
Using Data Converters with Adapter Services	6-39
Creating Custom Converters.....	6-39
Implementing Adapter Services and Business Rules	6-43
Implementing Import Services and Business Rules	6-43
Implementing Import Interval Services and Business Rules.....	6-45
Implementing Interval Services and Business Rules	6-47
Implementing COM Services and Business Rules.....	6-48
Implementing Assembly Services and Business Rules	6-51
Implementing CustomQuery Services and Business Rules	6-57
Implementing Rate Services and Business Rules	6-59
Implementing Oracle Utilities Meter Data Management Services and Business Rules	6-61

Business Rule Examples.....	6-62
Import Business Rule Example	6-62
Import Interval Business Rule Example	6-65
Interval Business Rule Example	6-68
COM Business Rule Example.....	6-70
Rate Business Rule Example.....	6-71
Assembly Business Rule Example.....	6-72
CustomQuery Business Rule Example.....	6-74
The Adapter Server	6-76
Set Up the LTMH.CFG.XML File.....	6-76
Starting the Adapter Server	6-76
The Adapter Monitor.....	6-77
Starting the Adapter Monitor.....	6-77
Runtime Services Tab.....	6-77
Runtime Queue Tab	6-78
Messaging Tool Tab	6-78
Securing and Encrypting the Adapter Server and Monitor	6-79
Technical Overview	6-79
Creating Certificates.....	6-79
Enabling Secure Communications	6-81

Chapter 7

Configuring Energy Information Platform Security	7-1
Security Features.....	7-2
Framework Features	7-3
User Preferences Features.....	7-4
Set Preferences Features	7-4
Options.....	7-4
Entity Management Features.....	7-5
Common Features	7-8
Securing Interval Data Sources	7-10
Work Queue Features.....	7-11
Work Queues Features.....	7-11
Securing Work Queue Items	7-12
Adapter Features	7-13
Data Source Features	7-14
Securing Reports.....	7-15
Overview	7-15
Securing the Contents of the Run Reports Screen	7-15
Securing the Contents of the View Reports Screen	7-16

Chapter 8

Setting Up Processing to run in Batch Mode.....	8-1
Batch Executables Overview.....	8-2
Data Utilities	8-2
Billing.....	8-3
Miscellaneous.....	8-3
Importing Data	8-4
Importing Customer Data	8-4
Importing Interval Data.....	8-9
Importing Enhanced Interval Data.....	8-26
Exporting Data	8-29
Exporting and Deleting Customer Data	8-29
Exporting and Deleting Interval Data	8-35
Executing Oracle Utilities Rules Language Rate Schedules.....	8-41
Purpose of RUNRS	8-41

RUNRS Inputs	8-41
RUNRS Command Syntax	8-42
Automated Profiling	8-48
Purpose of AUTOPROF.....	8-48
AUTOPROF Inputs and Other Prerequisites	8-48
AUTOPROF Command Syntax.....	8-50
AUTOPROF Processing.....	8-52
Migrating RCDATA.....	8-53
Purpose of RCTORDBM.....	8-53
RCTORDBM Command Syntax.....	8-54
Chapter 9	
Creating a CIS Transaction Record Output File.....	9-1
Transaction Type Description File Format.....	9-2
Output File.....	9-2
Control File	9-3
Sample CIS Transaction File	9-8
Print Detail.....	9-9
Chapter 10	
Customizing the Energy Information Platform	10-1
Overview.....	10-2
What Sort of Customization Is Possible?.....	10-2
Basic File and Directory Structure	10-2
Customization Ground Rules	10-3
A Note About Dictionary Files	10-3
Creating Custom Menus and Screens.....	10-4
Adding Custom Menu Items.....	10-4
Securing Custom Menu Items.....	10-7
Calling Existing Screens.....	10-13
Creating Custom Pages	10-20
Part Two	
System Administration.....	1-i
Chapter 11	
Oracle Utilities Energy Information Platform Services	11-1
Interval Data Storage and Access	11-2
Standard Interval Data Storage.....	11-2
Enhanced Interval Data Storage.....	11-12
Interval Data Access.....	11-21
Interval Data Versioning.....	11-21
Energy Information Platform Report Monitor	11-23
Energy Information Platform Report Scheduler.....	11-24
Energy Information Platform Logging Framework	11-25
Logging Framework Output File Format	11-25
Default Behavior	11-26
Merging Tracing and Logging Files.....	11-28
Tracing Levels.....	11-29
Database Auditing.....	11-36
Types of Database Auditing.....	11-36
Storing Auditing Information	11-36
Installing Auditing.....	11-37
Auditing Configuration and Administration.....	11-39
Auditing Reports	11-39

Chapter 12

Energy Information Platform Security	12-1
Overview	12-2
How Security Works.....	12-2
What Applications Are Secured?	12-2
Security Concepts and Data.....	12-3
Security Modes	12-3
Types of Security.....	12-3
Security Data.....	12-4
Working with Function-Level Security	12-7
Types of Function-Level Security	12-7
Applying Function-Level Security	12-7
Working with Record-Level Security	12-9
Types of Record-Level Security.....	12-9
Applying Record-Level Security	12-12
Examples of Using Record-Level Security.....	12-12
Limits on Record-level Security.....	12-13
The Security User Interface	12-14
Setting Up Security.....	12-15
Setting Up Security in Default Mode.....	12-15
Setting Up Security in LDAP Mode.....	12-20
Integrating Security with LDAP.....	12-23
Overview	12-23
Integration Details	12-23
Troubleshooting.....	12-26
Useful Tools.....	12-26
Connecting to LDAP Servers over SSL	12-27
Single Sign-On	12-29
LSSECURE.CFG.XML File	12-29
Creating Energy Information Platform User IDs for Single Sign-On	12-30
IIS Configuration	12-30

Chapter 13

Energy Information Platform Reporting Framework	13-1
Overview	13-2
Reporting Framework Components	13-2
How the Energy Information Platform Reporting Framework Works	13-4
Supported Report Formats.....	13-5
Energy Information Platform Reporting Framework Database Tables	13-6
Report Category	13-6
Report Instance	13-6
Report Instance Log.....	13-7
Report Priority.....	13-8
Report Template	13-8
Report Template Product.....	13-9
Report Type	13-10
Report Unit	13-10
The Report Administration User Interface	13-11
Running and Viewing Reports	13-12
Running Reports from the User Interface	13-12
Running Reports from the Oracle Utilities Rules Language.....	13-12
Running Reports in Batch Mode.....	13-12
Viewing Reports.....	13-15
Configuring the Energy Information Platform Reporting Framework for use with Oracle BI Publisher ..	13-16
Oracle BI Publisher Configuration	13-16

Energy Information Platform Configuration	13-19
Adding Oracle BI Publisher Reports to the Energy Information Platform.....	13-20
Designing Oracle BI Publisher Reports for use with the Reporting Framework	13-22
Report Folders.....	13-22
Report Data Sources.....	13-22
Creating Parameters.....	13-22
Adding Custom Crystal Reports to the Oracle Utilities Energy Information Platform.....	13-25
Adding Custom Reports Using Static ASP Pages.....	13-26
Designing Crystal Reports for use with the Reporting Framework	13-28
Report Data Sources.....	13-28
Creating Parameters and Parameter Fields	13-28
Adding Oracle Utilities Rules Language Rate Schedules to the Energy Information Platform.....	13-30
Adding Rate Schedules.....	13-30
Designing Rate Schedules for use with the Reporting Framework	13-31
Reporting on Interval Data.....	13-36
Creating Crystal Reports Templates.....	13-36
Converting Interval Data.....	13-36
Interval Data Reporting Tables	13-38
Chapter 14	
Troubleshooting.....	14-1
Rules Language Profiling.....	14-2
Running Trial Calculations	14-2
Within an Application	14-2
Sample Profile Output	14-4
SQL Tracing.....	14-5
Enabling SQL Tracing	14-5
Sample SQL Tracing File.....	14-5
Oracle Utilities Energy Information Platform Diagnostics	14-6
Oracle Utilities Energy Information Platform Adapter Logging.....	14-8
Setting the Debug Level.....	14-8
Viewing Log Files	14-8
Debug Level Details and Sample Log Files	14-9
Part Three	
COM Interfaces.....	1-i
Chapter 15	
Energy Information Platform COM Interfaces	15-1
The Energy Information Platform COM Interfaces	15-2
COM Interface Level Functions.....	15-2
Energy Information Platform Level Functions.....	15-3
How the Energy Information Platform COM Interfaces Work.....	15-4
The Energy Information Platform COM Interface Format.....	15-5
Basic Syntax	15-6
Common Argument Definitions.....	15-7
xmlDataSource	15-7
Date Formats	15-8
Date.....	15-8
Date-Time	15-8
Chapter 16	
Data Source Interface	16-1
Methods, Interface, and Syntax	16-2
Argument Definitions	16-4
Input Values	16-5
xmlDataSource	16-5

xmlQueryIn.....	16-7
Return Values.....	16-9
xmlQueryOut.....	16-9
Chapter 17	
Energy Information Platform Database Import Interface	17-1
Methods, Interface, and Syntax.....	17-2
Interface Arguments.....	17-3
Input Values	17-4
xmlImport	17-4
Return Values.....	17-6
xmlImportResult	17-6
Chapter 18	
Oracle Utilities Rules Language and Analysis Interface	18-1
Methods, Interface, and Syntax.....	18-2
Rules Language Interface Methods.....	18-2
Analysis Interface Methods.....	18-4
Interface Arguments.....	18-6
Input Values	18-8
xmlRulesSchedule	18-8
xmlAnalysis	18-11
Return Values.....	18-15
xmlSaveToXML.....	18-15
xmlAnalysisResult	18-16
Chapter 19	
Energy Information Platform Work Queues Interface.....	19-1
Methods, Interfaces, and Syntax	19-2
Interface Arguments.....	19-5
Input Values	19-6
xmlWQItem XML Elements	19-6
XML Examples - xmlWQItem	19-6
Element Descriptions - xmlWQItem	19-10
Return Values.....	19-13
xmlWQItemOut.....	19-13
Chapter 20	
Energy Information Platform Interval Data Interfaces.....	20-1
LSINTD Interface Methods, and Parameter Descriptions	20-2
IIntDDDataSource Interface	20-2
IIntDRecorderList Interface	20-5
IIntDRecorder Interface.....	20-6
IIntDChannelList Interface.....	20-6
IIntDChannel Interface	20-7
IIntDCutSeries Interface	20-8
IIntDCutList Interface	20-8
IIntDCutMessageList Interface	20-8
IIntDCutMessage Interface.....	20-9
IIntervalList Interface	20-10
IIntDPureCut Interface	20-10
IIntDCut Interface.....	20-14
IIntDCutValue Interface	20-20
IIntDProvider Interface.....	20-21
Interval Data Source Providers	20-22
RDBProvider.....	20-22
BTEProvider.....	20-22

DATASOURCE Definition.....	20-23
Sample Application using LSINTD	20-24
INTDVB Compatibility	20-30
Part Four	
Appendices	1-i
Appendix A	
Oracle Utilities Data Repository Database Schema.....	A-1
Energy Information Platform Schema p. 1	A-2
Energy Information Platform Schema p. 2	A-3
Energy Information Platform Schema - Security	A-4
Energy Information Platform Schema - Reporting	A-5
Energy Information Platform Schema - Messaging.....	A-6
Energy Information Platform Schema - Work Queues	A-7
Energy Information Platform Schema - Interval Data Manager	A-8
Energy Information Platform Schema - Adapter.....	A-9
Energy Information Platform Schema - Service Points and Market Participants.....	A-10
Energy Information Platform Schema - Web Services	A-11
Appendix B	
Oracle Utilities Data Repository Schema Import File Format	B-1
Record Formats	B-2
A Note Regarding Dependent Records.....	B-2
Oracle Utilities Import Format (*.imp).....	B-3
Rules.....	B-5
Date Formats	B-5
Oracle Utilities XML Import Format (*.xml)	B-6
DTD - Oracle Utilities XML Import Format.....	B-6
XML Examples - Oracle Utilities XML Import Format	B-6
Element Descriptions - Oracle Utilities XML Import Format.....	B-7
Appendix C	
Oracle Utilities Enhanced Input/Output Interval Data Format	C-1
Enhanced Format.....	C-2
General Field Descriptions.....	C-2
Field Relationships and Requirements.....	C-3
Enhanced Format Details	C-12
Sample Files.....	C-15
Enhanced Direct Input File	C-15
Standard Direct Input File.....	C-15
Units of Measure	C-16
Appendix D	
Oracle Utilities Standard XML Interval Data Format.....	D-1
Standard XML Format	D-2
General Node Descriptions	D-2
General Rules.....	D-3
Node Relationships and Requirements	D-4
Interval Values.....	D-12
XML Standard File Format Example.....	D-14
XML Standard File Format DTD.....	D-15
XML Compact Format.....	D-16
XML Compact File Format Examples	D-16
XML Compact File Format Element and Attribute Descriptions.....	D-17
Interval Values.....	D-19

Appendix E

Oracle Utilities Comma Separated Values (CSV) Interval Data Format E-1

 CSV Format E-2

 General Description E-3

 Document Description Record E-4

 Field Relationships and Requirements..... E-4

 Format Details E-13

 Sample CSV File E-16

 Differences with the .LSE format..... E-16

 Field Delimiters..... E-16

 Field Types..... E-16

 Document Description Record E-16

 Date, Time, and Decimal Value Representation in Records..... E-16

 Interval Data Records E-16

 Compatibility Issues E-17

Index

What's New

New Features in the Oracle Utilities Energy Information Platform Configuration Guide

This chapter outlines the new features of the 1.6.0.0 release of the Oracle Utilities Energy Information Platform that are documented in this guide.

New Features for Release 1.6.0.0

Feature	Description	For more information, refer to...
Support for Oracle Business Intelligence Publisher	This release includes support for publishing reports using Oracle Business Intelligence Publisher version 10.1.3.4 or 11.1.1 (as of v1.6.1.7).	LSREPORTMONITOR.CFG.XML on page 2-35 Chapter 13: Energy Information Platform Reporting Framework , including: <ul style="list-style-type: none">• Configuring the Energy Information Platform Reporting Framework for use with Oracle BI Publisher on page 13-16• Adding Oracle BI Publisher Reports to the Energy Information Platform on page 13-20• Designing Oracle BI Publisher Reports for use with the Reporting Framework on page 13-22
New Interval Data Converters	This release includes new interval data converters that can be used when importing interval data using the Adapter.	Chapter 6: Setting Up, Configuring, and Running the Energy Information Platform Adapter , including: <ul style="list-style-type: none">• IntDExpConverter on page 6-32
Enhanced Tracing and Logging	This release includes enhancements to the tracing and logging functionality of the Energy Information Platform Logging Framework.	LSLOGGER.CFG.XML on page 2-27 Energy Information Platform Logging Framework on page 11-25

New Features for Release 1.6.1.0

Feature	Description	For more information, refer to...
Utility to set administrator password	The LSSecureInit.exe program has been enhanced to support setting the password for the administrator login (“admin”).	Set Administration Login Password on page 12-16
Support for securing Adapter Server and Adapter Monitor communications	Communications between the Adapter Server and Adapter Monitor can now be secured and encrypted via SSL.	Securing and Encrypting the Adapter Server and Monitor on page 6-79

New Features for Release 1.6.1.17

Feature	Description	For more information, refer to...
Include Data Manager memory profile snapshot in code profiles	When creating code profiles using the Trial Calculation module, the memory profile of the Data Manager application can be included in the code profile.	See LODESTAR.CFG on page 2-2 and Rules Language Profiling on page 14-2
Include Adapter Runtime service memory profiles in Adapter log files.	Adapter log files can contain periodic logging of the number of running Adapter Runtime Services and their individual memory profiles.	See LTMH.CFG.XML on page 2-46 and Including Runtime Service Memory Profiles on page 14-8
Include server / runtime service network communication logging in Adapter log files	Network communication operational logging can be included in the Adapter/ Transaction Management log files.	See LTMH.CFG.XML on page 2-46 and Adapter Server and Runtime Service Network Communication on page 14-8

Chapter 1

Overview

This chapter provides an overview of the installation and configuration of the Oracle Utilities Energy Information Platform, including:

- **Configuration Overview**
- **What is this book?**
- **International and Multiple Currency Support**
- **Notes, Cautions, and Warnings**

Configuration Overview

Configuring the Oracle Utilities Energy Information Platform involves the following steps:

- Set up Energy Information Platform database records as described in **Chapter 3: Setting Up Energy Information Platform Database Records**.
- Set up interval data and related records for use with Interval Data Viewer and Interval Data Manager as described in **Chapter 4: Configuring Interval Data Viewer and Interval Data Manager**.
- Set up and configure work queues and work queue types used by the Work Queues applications as described in **Chapter 5: Configuring Work Queues**.
- Set up other Oracle Utilities applications to send messages to the Work Queues application as described in **Chapter 5: Configuring Work Queues**.
- Set up and configure the Adapter as described in **Chapter 6: Setting Up, Configuring, and Running the Energy Information Platform Adapter**.
- Start the Adapter Server to run the Adapter as described in **Chapter 6: Setting Up, Configuring, and Running the Energy Information Platform Adapter**.
- Set up and configure security for use with the Energy Information Platform as described in **Chapter 7: Configuring Energy Information Platform Security**.
- Set up processes to run in batch mode as described in **Chapter 8: Setting Up Processing to run in Batch Mode**.
- Customize the Energy Information Platform user interface as described in **Chapter 10: Customizing the Energy Information Platform**.
- Set up and configure interfaces to the Energy Information Platform from external systems and applications. Available COM interfaces are described in **Part Three: COM Interfaces**

What is this book?

This book describes how to install and configure the Oracle Utilities Energy Information Platform, including the following:

- **Chapter 1: Overview** (this chapter) provides an overview of configuring the Energy Information Platform.
- **Chapter 2: Configuration Files** describes configuration files used by the Energy Information Platform.
- **Chapter 3: Setting Up Energy Information Platform Database Records** describes how to set up and configure database records used by the Energy Information Platform.
- **Chapter 4: Configuring Interval Data Viewer and Interval Data Manager** provides an overview of setting up and configuring Interval Data Viewer and Interval Data Manager.
- **Chapter 5: Configuring Work Queues** provides an overview of the database tables used by the Work Queues application, the actions that can be performed on work queue items, how to set up work queues and work queue types, and how to set up Oracle Utilities applications to send work queue items to the Work Queues application.
- **Chapter 6: Setting Up, Configuring, and Running the Energy Information Platform Adapter** describes how to set up and configure the Adapter, as well as how to start and run the Adapter Server and Adapter Monitor used by the Adapter.
- **Chapter 7: Configuring Energy Information Platform Security** describes how to configure security for use with the Energy Information Platform.
- **Chapter 8: Setting Up Processing to run in Batch Mode** describes how to set up processes such as importing/exporting data in batch mode.
- **Chapter 9: Creating a CIS Transaction Record Output File** describes how to create CIS output files from Oracle Utilities Rules Language.
- **Chapter 10: Customizing the Energy Information Platform** describes how to customize the Energy Information Platform user interface.
- **Chapter 11: Oracle Utilities Energy Information Platform Services** describes services and features of the Energy Information Platform.
- **Chapter 12: Energy Information Platform Security** describes Energy Information Platform security administration.
- **Chapter 13: Energy Information Platform Reporting Framework** describes the Energy Information Platform reporting framework.
- **Chapter 14: Troubleshooting** provides information related to troubleshooting performance issues.
- **Chapter 15: Energy Information Platform COM Interfaces** describes the COM interfaces available with the Energy Information Platform.
- **Chapter 16: Data Source Interface** describes the Energy Information Platform Data Source COM interface, used to access the Oracle Utilities Data Repository.
- **Chapter 17: Energy Information Platform Database Import Interface** describes the Energy Information Platform Database Import interface, used to import data into the Oracle Utilities Data Repository.
- **Chapter 18: Oracle Utilities Rules Language and Analysis Interface** describes the Energy Information Platform Rules Language interface, used to execute Rules Language processing.

- **Chapter 19: Energy Information Platform Work Queues Interface** describes the Energy Information Platform Work Queues interface, used to create and maintain work queue items in the Oracle Utilities Data Repository.
- **Chapter 20: Energy Information Platform Interval Data Interfaces** describes the Energy Information Platform Interval Data interface, used to provide access to interval data stored in the Oracle Utilities Data Repository.
- **Appendix A: Oracle Utilities Data Repository Database Schema** provides diagrams of the Oracle Utilities Data Repository database schema.
- **Appendix B: Oracle Utilities Data Repository Schema Import File Format** describes the Oracle Utilities Import/Export file format (IMP).
- **Appendix C: Oracle Utilities Enhanced Input/Output Interval Data Format** describes the Enhanced Oracle Utilities Input/Output file format (LSE) used for interval data.
- **Appendix D: Oracle Utilities Standard XML Interval Data Format** describes the Standard Oracle Utilities XML interval data file format.
- **Appendix E: Oracle Utilities Comma Separated Values (CSV) Interval Data Format** describes the Oracle Utilities Comma Separated Values (CSV) interval data file format.

International and Multiple Currency Support

This section outlines international and multiple currency support in the Oracle Utilities Energy Information Platform, including:

- **Locales and Languages**
- **Date and Numeric Presentation**
- **Currency Presentation**
- **Multiple Currency Support**
- **File Formats**
- **Database Support**

Locales and Languages

Two fundamental concepts underlying international support in Oracle Utilities applications are **languages** and **locales**. Languages determine the specific language in which text labels and controls of an application appear. Locales determine the specific region or country in which the user is working. Specifically, locales are used to specify display and input formatting for:

- Date/time and numeric presentation, and
- Currency presentation

How a user selects a language and/or locale is determined by the type of application being used.

Web-enabled Products

For web-enabled products, a user's locale and language are selected from the **Locales** tab of the **Set Preferences** screen. The selected locale and language are stored in the user's security properties. The available locales on this screen are specified in the LOCALES.CFG.XML configuration file (see **Chapter 2: Configuration Files** in the *Oracle Utilities Energy Information Platform Configuration Guide* for more information). Locales included in the LOCALES.CFG.XML file must also be installed on the web server and be accessible from the Regional Settings Control Panel. If a selected locale is not installed on the web server, currency formatting and presentation may be incorrect.

If a user does not select a locale (or selects a locale of "None"), the default locale is the Server System Locale (defined by the operating system installed on the web server). Oracle Utilities recommends that all users specify a locale before using internationalized applications.

Client/Server Products

For client/server products, the user's locale is determined by the Regional Settings of the client and/or application server machine.

Languages

The language for client/server products is specified using a command line switch and/or a parameter in the LODESTAR.CFG file. See **Oracle Utilities Application Command Line Parameters** on page 4-10 in the *Oracle Utilities Energy Information Platform Installation Guide* for more information about command line parameters. See **LODESTAR.CFG** on page 2-2 in the *Oracle Utilities Energy Information Platform Configuration Guide* for more information about the LODESTAR.CFG file. At this time, available languages include English, French, German, and Italian.

Reports

For client/server applications, the locale for reports is determined by the Regional Settings of the machine on which the report is run.

For web-enabled applications, the locale for reports is based on the Server System Locale (defined by the operating system installed on the web server)

Languages

For client/server applications, the language that appears on reports is determined by the language of the application that generated the report, specified either by a command line parameter or a configuration file parameter.

For web-enabled applications, the language is specified by the Regional Settings of the machine on which the report is run, unless overridden by the LANGUAGE parameter in the LODESTAR.CFG file.

Date and Numeric Presentation

The display and input of dates and numbers is based on the selected locale.

Web-enabled products

Display formatting of both numbers and date/times is based on the user's selected locale.

Copy, cut, and paste operations are supported for all input controls. Required fields are identified by asterisks. Null values are represented by blank (empty) controls. Moving the mouse pointer over an input area displays sample formatting for that field.

Number input controls only accept formatting based on the user's selected locale. The only allowable characters include digits, decimal separator, and minus sign. Any other characters are filtered from input.

Date input controls only accept formatting based on the user's selected locale. Dates can either be keyed in or selected from a pop-up calendar control. All times are keyed in and based on 24 hour format (no AM or PM indicator).

Input validation occurs upon data entry.

Client/Server Products

Formatting for numbers and date/times is based on the locale determined by the regional settings of the client and/or application server machine. Date/time formatting can be overridden from the General Options tab of the Default Options dialog.

Reports

Display formatting of both numbers and date/times for reports is based on the Server System Locale on the machine on which the reports are run.

Currency Presentation

Currency information is stored in the LSCurrency table in the Oracle Utilities Data Repository. This table contains one or more currencies, each defined by a currency code, currency symbol, and formatting rules (such as the number of decimal places). Currency codes in this table use ISO-4217 codes. The LSCurrency table also indicates the Default currency for the database. See **LSCurrency** on page 3-3 in the *Oracle Utilities Energy Information Platform Configuration Guide* for more information about setting up the LSCurrency table.

Note: Although records in the LSCurrency table are not required, Oracle Utilities strongly recommends that at least one record with all properties specified be created and set as "Default". This will ensure consistent display of currency information.

Currency Presentation in Oracle Utilities Receivables Component and Oracle Utilities Quotations Management

Currency input and display is identical to that for numbers with two exceptions:

- A currency indicator is also displayed next to currency input controls and currency displays. The specific currency symbol displayed and the manner in which currency is formatted are determined in the following order:
 - a. The currency associated with the account being viewed (from the LSCurrency field in the Account table).
 - b. The default currency in the LSCurrency table. If there is no default currency, the first record in the LSCurrency table (sorted alpha-numerically by Currency Code).
 - b. If the LSCurrency table is not present or does not contain any records, the currency and formatting of the Server System Locale (defined by the operating system installed on the web server), specified in the **HKEY_USERS\DEFAULT\Control Panel\International** key in the Windows Registry on the web server.
- Currency formatting rules for the user's selected locale are overridden if specific formatting rules for the currency are stored in the LSCurrency table. Typically, the only formatting rules that would be overridden at the currency level would be the ISO code, symbol, and number of decimal places. This applies to Oracle Utilities Receivables Component reports as well, except that the initial locale formatting rules come from the regional settings of the machine on which the report is run.

Currency Presentation in Other Web-enabled Products

Currently, other web-enabled products do not require input of currency values.

Display of currencies is based on the formatting of the selected locale and of the default currency in the database (currency specific formatting always overrides locale specific formatting).

Enabling Display of International Currency Symbols

Display of international currency symbols requires the appropriate Encoding option is selected on Internet Explorer.

How to select encoding options:

1. Select **View→Encoding** from the Internet Explorer menu bar.
2. Select the appropriate option from the list. For example, to enable displaying of the Euro symbol and other Western European currency symbols, select the **Western European (Windows)** option.

Currency Presentation in Client/Server Products

Client/server products do not require input of currency values.

Display of currencies is based on the formatting of the client machine's regional settings and of the default currency in the database. Currency specific formatting always overrides locale specific formatting. If the default currency in the database does not specify any formatting, the currency symbol is used for formatting. If the currency table does not exist or does not contain any records then the default currency is determined by the Regional Settings on the client/server machine. This applies to reports as well.

Multiple Currency Support

Multiple Currency Support allows for accounts with different associated currencies. This is accomplished through the presence of the Currency Code column on the Account, Journal Account, and Checking Account tables in the Oracle Utilities Data Repository. This column references records in the LSCurrency table

Multiple Currency and Oracle Utilities Receivables Component

All input screens which have amount fields on them indicate what currency type is expected as input.

All elements that represent a currency in Oracle Utilities Receivables Component XML structures have a CURRENCY attribute that indicates the corresponding currency type for the amount. The string values used for this attribute are the same as the Currency Code values in the LSCurrency table. On output, the CURRENCY attribute is set for the account as specified in the database. If the account's currency type is not specified in the database, the default currency is used.

A CURRENCY attribute (using the standard stem.tail notation) is supported for Oracle Utilities Receivables Component transaction identifiers to specify the currency type for the account.

On input, the CURRENCY attribute of all Oracle Utilities Receivables Component transactions is checked against the currency value of the account against which the transaction is being posted using the following rules:

1. If a currency type is specified for the account in the database, the transaction must indicate the same currency type or it is an error.
2. If a currency type is not specified for the account in the database, the transaction must indicate either no specific currency type or the default currency type for the database or it is an error.

All transactions that involve two or more accounts (such as Payment and Balance Transfers) validate that the currency types are the same for each account (no currency translation will occur). The credit/debit account pairs for each journal translation record are validated to ensure that they both have the same currency type. Only journal translation records that have credit/debit accounts with the same currency type as the account against which the transaction is being posted are used. Additional currency validations occur for payment files and refund checks.

Multiple Currency and Oracle Utilities Receivables Component Reports

Where applicable, Oracle Utilities Receivables Component reports are grouped by Currency Code.

Multiple Currency and Other Oracle Utilities Products

Oracle Utilities Billing Component, Oracle Utilities Rate Management and Data Manager reports display the Currency Code associated with each account where applicable. Reports that contain multiple currency types do not include the Summary Results page.

In Oracle Utilities Rate Management - Revenue Forecasting, the default currency type is used for all presentation of currency amounts, regardless of whether or not a specific currency has been indicated in the database for any accounts.

File Formats

Input/Output File Formats

Numbers and date/times in input and output files (such as files used with the PLIMPORT program) are independent from locale settings.

Non-English characters are supported in input/output files.

XML Formats

All dates, times, and numbers contained in XML files are represented using the standard formatting rules as defined by the W3C XML Schema Recommendation. For backward compatibility, previously accepted formats for dates and times will continue to be accepted on input, however, all output will use the XML Schema representation.

Note: XML files created by the SAVE TO XML Rules Language statement are an exception. The date, time, and number formats in Rules Language generated XML files are based on the Rules Language configuration used to create them.

Non-English characters are supported in XML files

Database Support

In addition to the LSCurrency table and related columns in the Account, Journal Account, and Checking Account tables, database support for internationalization includes the addition of the COUNTRY column in the Address table and the change from ZIP Code to Postal Code on user interfaces.

Notes, Cautions, and Warnings

The following notes, cautions, etc. apply to all Oracle Utilities products. In the development of these applications tradeoffs were made in favor of performance, as opposed to functionality. The following explains some of these tradeoffs.

- All Oracle Utilities products accept dates and bill months from January 1, 1901 through December 31, 3000.
- All Oracle Utilities products use the C data type “double” to store floating point values. This is limited to 15 digits of precision. Strings or numbers with more digits are rounded to the nearest value supported by doubles when they are converted to a float.
- Use of the ampersand (“&”) symbol in ID fields results in errors when exporting data in XML format.
- If the database Browser (Account Info, Customer/Account(s) or Customer Database) is running and a database Import is run, the Import may hang. The Browser may have database “pages” locked, which will hang Import if it tries to update one of these pages (the Import will continue when the Browser is closed). In general, you should close Browser windows before running the Import function.
- If the Import function deletes records that are on view in a Browser, the Browser may display an error message when it attempts to read the deleted record. Again, close Browser windows before running the Import function.
- Some data is cached for faster retrieval. This includes operating company codes and names, determinant information, and other relatively static data. If these values are changed in the Browser or via the Import program, the cache will not be updated (and the new values visible) in an application until the application restarts or the user selects Edit - Refresh Cached Tables from the menu.
- When most analysis dialogs are cancelled they are actually hidden, so that all the selections remain. When the analysis is selected again, the dialog displays the selections from the last time it was viewed. To refresh such a dialog, use the system Close menu pick or the X pick in the upper right corner. This destroys the dialog and re-initializes a new one.
- The maximum length of all codes and names is 64 characters. However, to keep dialog boxes at a reasonable size, a display maximum display width of 25 characters is assumed. If you use longer names, some displays will be distorted.
- Oracle Utilities Rate Management and Oracle Utilities Billing Component have some differences in how they perform their revenue or bill calculations. All Oracle Utilities Rate Management calculations use the dates in the BillHistory records; in Oracle Utilities Billing Component the interval data cut dates may override the BillHistory dates. In both cases, the Save Statement will write data to the database; this data is “rolled back” when the analysis completes. However, in a Oracle Utilities Billing Component analysis the data is actually saved when the bill is approved.
- Oracle Utilities Billing Component supports multiple Bill History records in a bill month. Oracle Utilities Rate Management supports only one Bill History record in a bill month.
- Oracle Utilities Billing Component requires the ReadDates in all Bill History records to be not NULL; Oracle Utilities Rate Management allows NULL ReadDates in Bill History records.
- Oracle Utilities Rate Management does “the best it can”. If a historical value is requested, the nearest value will be used if the requested one does not exist. Oracle Utilities Billing Component is exact, with these historical function exceptions:
 - If HISTVALUE is used to request a value past the end of history values, 0 is returned.

- If the start month to historical functions is 1 and there is only the current value, 0 is returned. However, if the start month is greater than 1 and greater than the number of historical values, it is an error.
- If the stop month is greater than the number of historical values, it is set to the last period with data.
- When a rate schedule is used for an account and the analysis is Automatic Billing, Approval Required billing, or Current/Final Bill, the rate schedule must have either the \$EFFECTIVE_REVENUE identifier or another identifier set as the total through the REVENUE TOTAL statement. This guarantees that there is a total revenue.
- In any numeric edit field, to make the value negative press the minus (-) key when the cursor is left of any decimal point. To make a negative value positive, press the plus (+) key. Note that you cannot delete or backspace over a minus sign to remove it.
- If you use an OR conjunction in a Prompted Query, the actual query may run **very slowly**. This is a limitation in all relational databases.
- If you change the Windows Regional Settings while an application (Data Manager, Oracle Utilities Billing Component, etc.) is running, you must restart the application in order for the new settings to take effect.

Part One

Configuration

Part One describes configuration of the Oracle Utilities Energy Information Platform, and contains the following chapters:

- **Chapter 2: Configuration Files**
- **Chapter 3: Setting Up Energy Information Platform Database Records**
- **Chapter 4: Configuring Interval Data Viewer and Interval Data Manager**
- **Chapter 5: Configuring Work Queues**
- **Chapter 6: Setting Up, Configuring, and Running the Energy Information Platform Adapter**
- **Chapter 7: Configuring Energy Information Platform Security**
- **Chapter 8: Setting Up Processing to run in Batch Mode**
- **Chapter 9: Creating a CIS Transaction Record Output File**
- **Chapter 10: Customizing the Energy Information Platform**

Chapter 2

Configuration Files

This chapter provides guidelines for modifying a number of configuration files used by Oracle Utilities applications. These files are used to customize the working environment of your Oracle Utilities application software. The configuration files described in this chapter include:

- **LODESTAR.CFG**: Sets parameters used by Oracle Utilities applications.
- **EXTENDEDSTATUSES.CFG.XML Configuration File**: Defines extended interval data status codes.
- **IDV.CFG.XML**: Defines behavior of Interval Data Viewer and Interval Data Manager features.
- **INTDCONFIG.CFG.XML**: Defines interval data behavior, including how data is imported and exported, as well as how interval timestamps are displayed.
- **LOCALES.CFG.XML**: Specifies regional locales available to the web-enabled Customer Choice Suite.
- **LSCALENDAR.CFG.XML**: Defines DST rules for specified regions.
- **LSDB.CFG.XML**: Defines a log file used to capture all ODBC statements (queries) executed by Oracle Utilities applications.
- **LSLOGGER.CFG.XML**: Specifies parameters for generating log files from Oracle Utilities applications.
- **LSRELAY.CFG.XML**: Defines the SMTP server used to send email messages from the Energy Information Platform.
- **LSREPORTMONITOR.CFG.XML**: Specifies parameters for running reports from the web-enabled Customer Choice Suite.
- **LSSCHDLR.CFG.XML**: Defines the data source(s) monitored by the Oracle Utilities Schedule Service (LSSCHDLR.exe).
- **LSSECURE.CFG.XML**: Specifies the data source used by the Security Administration tools.
- **LTMH.CFG.XML**: Specifies the database used by the Oracle Utilities Adapter and the Oracle Utilities Transaction Management.
- **Tools Files (*.TLS)**: Allows you to include other Oracle Utilities products and third-party software on the Tools Menu of your Oracle Utilities application software.

This chapter also includes a description of a utility that can be used to encrypt connection string and other information stored within XML configuration files. See **Encrypting Configuration Files** on page 2-54 for more information.

LODESTAR.CFG

LODESTAR.CFG is a text file used to customize the working environment of the Oracle Utilities application software. Though this file is not required to run the Oracle Utilities application software in most circumstances, its use is recommended to make the working environment settings explicit.

A sample of this file is provided in the **C:\LODESTAR\CFG\Examples\CFG** directory.

Unless specified otherwise in a command line, the LODESTAR.CFG file should be located in the C:\LODESTAR\CFG directory.

The three possible configurations for the Oracle Utilities application software and their requirements for the LODESTAR.CFG file are as follows:

1. **Stand-alone workstation:** The Oracle Utilities Data Repository, the executables, and the data files all reside on the same end-user workstation. This does not require a LODESTAR.CFG, unless the following special instructions are needed:

```
USERDIR = C:\LODESTAR\USER
```

```
RCDATADIR = C:\LODESTAR\RCDATA
```

2. **Shared Network Database:** The Oracle Utilities Data Repository is installed on a server; all users share the same database, but the application software is installed locally on each end-user machine. This generally speeds up the loading of executables, and provides faster disk access. This does not usually require a LODESTAR.CFG.

```
USERDIR = C:\LODESTAR\USER
```

```
RCDATADIR=\\<MACHINE_NAME>\<RESOURCE>\LODESTAR\<DATASOURCE>
```

This allows sharing of RCDATA files.

3. **Shared Network Database and Application file:** In this case, the Customer Database is installed on a Server and the application software is installed on a network File Server. To invoke the application, each end-user machine must be connected to the File Server, and must have the configuration file properly set up. The configuration file can reside either on the same file server as the application, or on each end-user machine. Care must be taken when using the former approach to make sure the end-users have the same directory structure:

```
USERDIR = C:\LODESTAR\USER, or
```

```
USERDIR = N:\LODESTAR\USER
```

```
LOGDIR = C:\LODESTAR\USER*
```

```
RCDATADIR=\\<MACHINE_NAME>\<RESOURCE>\LODESTAR\<DATASOURCE>
```

***Note:** It is important to have this separate from USERDIR when the USERDIR is shared on the network.

Allowable Settings

General/Database Settings (with default values):

DATEFORMAT = The date format string for your database. The string uses the format items described at the end of this list.

DBREQUEST_TIMEOUT = The number of seconds that a database request will wait for a response before returning an error. This is used to override the ADO default value of 30 seconds.

ENHANCED_PRECISION_SWITCH: If present, the full value of decimal values is displayed (up to 15 digits of precision), instead of being rounded off to six decimal places (the default behavior).

LANGUAGE = XXX, where XXX is a three letter language abbreviation that specifies the language displayed by the application. Supported languages include DEU (German), ENU (American English), FRA (French), and ITA (Italian). If the language is set both in the configuration file and in the command line, the configuration file takes precedence.

Note: This parameter overrides the Regional Settings and Locale selection when running reports from web-based applications.

LOGDIR = C:\LODESTAR\USER: where to place the runtime log files generated by the application software. If not specified, it defaults to the value in USERDIR. In a network application file server scenario, this value must point to different places for each user to allow concurrent use; otherwise, only one user can run the application at a time.

LOWERCASE_USERID: If present, the entered userid is converted to all lowercase characters. This allows the user to type an uppercase or lowercase userid if the database ignores the case of the userid, and still get the user's correct Options. If this parameter is set, the user's Options will need to be reset if they were saved under a mixed case or uppercase userid.

Note: UPPERCASE_USERID takes precedence over LOWERCASE_USERID if both are present.

MAX_DBVALUES_AMOUNT = N. This indicates the maximum number of database values that will be reported when processing a rate schedule. Note: use of this parameter does not limit the number of records saved to the database. It only limits the number of records reported.

NORCDATAQUAL = Define as anything to turn off appending the qualifier to the RCDATADIR directory (as a subdirectory).

PLDBQUAL = The new default database qualifier. The original default is PWRLINE. If a qualifier is entered (here, at the command line, or via selection in the logon dialog box), the qualifier is appended to the RCDATADIR directory (as a subdirectory). All data files for the qualified tables will be stored in this subdirectory. The maximum length of a qualifier is seven characters. All qualifiers used in the database must be uppercase.

PLDBQUALIFIERS = A comma-separated list of valid database qualifiers. They will be shown in the logon dialog, if the command line does not set the qualifier, or specifies the logon dialog. If the user selects a qualifier from this list, the RCDATADIR will be set to a subdirectory of the value above, where the subdirectory name is the selected qualifier. The maximum length of a qualifier is seven characters. All qualifiers used in the database must be uppercase.

PLDBQUALNODEF = 1 If set to 1, the default database qualifier selection <none> will be listed.

RCDATADIR = C:\LODESTAR\RCDATA: where to find and place shared application data files. If not specified, it defaults to the ..\RCDATA directory parallel to where the application software resides.

SET_ENLIST_TO_FALSE: If this keyword is present, the "enlist=false" attribute is added to the database connection string. For more information about this attribute, refer to the *Oracle Data Provider for .NET Developer's Guide*.

UPPERCASE_USERID: If present, the entered userid is converted to all uppercase characters. This allows the user to type an uppercase or lowercase userid if the database ignores the case of the userid, and still get the user's correct Options. If this parameter is set, the user's Options will need to be reset if they were saved under a mixed case or lowercase userid.

Note: UPPERCASE_USERID takes precedence over LOWERCASE_USERID if both are present.

USERDIR = C:\LODESTAR\USER: where to find and place USER files. If not specified, it defaults to the ..\USER directory parallel to where the application software resides.

Interval Data Settings:

BATCH_SIZE = N. This indicates the number of imported cuts processed before the cuts are committed to the relational database. This should not be used when running more than one instance of the application.

DO_NOT_USE_UOM_FOR_SCALING: If this keyword is present, interval data is totaled when scaling or aggregating, regardless of the Totalize Method flag specified in the Unit-of-Measure table.

INTD_ALWAYS_GET_DSTTOTAL_AND_DSTENERGY: When present, values for DSTTOTAL and DSTENERGY are calculated for the fall Daylight Saving Hour for interval data handles that have their DST_PARTICIPANT flag set to "Y".

INTDBLOCKOP_DO_NOT_COMBINE_STAT: If this keyword is present, the INTDBLOCKOP function will not combine statistics when performing block operations on interval data handles.

INTD_BTE_DATA_COMPRESSION = Any value turns on data compression when a BTE file is created. Otherwise compression is off.

INTD_COUNT_MISSING: If this keyword is present, in the event of an undercount (in which there are fewer intervals than expected based on the cut's start and stop times and SPI) when loading interval data in the Rules Language function, missing intervals (value '0', status '9') will be added to fill the cut.

INTDDEFAULTDST = One of the letters N or Y (No or Yes). If the DSTPARTICIPANT value is not specified in an interval data Import file, it is set to this value, if present.

INTD_EXPORT_LINE_CT = Any positive number. Default is 6, minimum is 1, maximum is 120. Indicates the number of intervals to put on a line when exporting in new (LSE) format.

INTD_IMPORT_STOP_TIME_GAP_IS_ERROR: This indicates that for *.LSE or *.INP files, it is an error if there are not enough data values to fill all intervals from the start time through the stop time. For example, a one month cut of half-hour data should contain 1440 intervals. Loading a one month cut of half-hour data with only 1436 data values would be an error if this parameter is included. In addition, for .INP files, trailing missing values in a line are removed before this is checked. The default is to fill the gap with missing values (status code '9', value 0).

INTDJOINTIMESTAMP: If present, the Rules Language will load overlapping cuts based on a combination of the latest Start Time and the latest timestamp of the loaded cuts. See **Loading Overlapping Cuts** on page 7-20 in the *Oracle Utilities Rules Language User's Guide* for more information about loading overlapping cuts using the Rules Language.

INTDLOADLIST_STOP_ON_LIST_ERROR = 1 If present, it is an error if a list used by the INTDLOADLIST function does not exist.

INTD_NEW_JOIN = 2: If present, turns on the Interval Data Merge tab on the Default Options window (opened by selecting **Tools->Options**). See Interval Data Merge Options in the *Data Manager User's Guide*, and the description of the INTDJOIN Rules Language function in the *Oracle Utilities Rules Language User's Guide*, for more information.

INTD_XML_EXPORT_NO_MILLISECONDS = 1: Indicates that when exporting interval data in XML format using the INTDEXPORT function, datetime columns in the exported data will not contain milliseconds.

INTDVALUEMAX = Number. The number format is digits, an optional decimal point, more digits, and an optional “e” followed by digits; for example, 1.5e10 (1.5 to the power 10). The absolute value of each interval data value is compared to this number. If the value is larger than this number it is set to this number, and, if its status code is better than 'L', its status code is set to 'L'. The default maximum value is 10e20 (10 to the power 20).

INTDVALUEMIN = Number. The number format is digits, an optional decimal point, more digits, and an optional “e” followed by digits; for example, 2.3e-10 (2.3 to the power 10). The absolute value of each interval data value is compared to this number. If the value is between 0 and this number it is set to this number, and, if its status code is better than 'L', its status code is set to 'L'. The default minimum value is 10e-20 (10 to the power -20).

MV90MAPFILE: Specifies the path and file name of the MV90 map file used when importing MV-90 data.

MV90_CM_SIC_PI: This is used with MV90_STARTTIME_1. If this is set to anything (on) and the characters "PI" are the first two characters in the CM_SIC field in the customer record, then the data contains a leading partial interval and 60 seconds is **not** subtracted from its start time. This is for meters that set the start of a partial interval to the previous cut's stop time.

MV90_IMPORT = 1: If present, enables import of interval data in MV90 format.

MV90_INSERT_OVERWRITE_USING_DEFAULT: MV90 map files can contain values to be written to records in the Oracle Utilities Data Repository. Normally these values are used if the corresponding values in the original record are NULL. Setting this parameter will override this such that the values from the MV90 map file will overwrite the values from the original record, even if they are non-NULL.

MV90_REC_SPACE_TO_UNDERSCORE If this is set to anything (on), when importing interval data file in mv90 format (*.mv9), all non-leading and non-trailing spaces are converted to underscores.

MV90_STARTTIME_1=OFF: Disables the default behavior for MV90 interval data files (*.mv9), in which 60 seconds are subtracted from cut's start time.

NO_DEFAULT_MASK: When this keyword is present, interval data handles whose values consist of only zeros (0) and ones (1) are treated as regular interval data handles rather than as masks. Only handles created by mask functions are treated as masks.

USE_VMSGs = 1: Indicates that the LSCHANNELCUTVMSGs table is to be queried every time an interval data cut is loaded. If this parameter is not specified (or is set to a different value), the LSCHANNELCUTVMSGs table will not be queried, resulting in improved performance.

XMLEXPORTINTDSTARTTIME = 1 Indicates that if interval data is exported in Oracle Utilities Standard XML File Format (*.xml), the start time for each interval value will be included.

Report Settings:

BILL_CALC_DISTRIBUTION_LABEL = Sets the label of the Distribution column in the Bill Calculation section. Default is Distribution. If no value is supplied, the Distribution column is not displayed.

CUSTOM_FIELD_SEP = The separator between the label and the value in the header part of reports. The separator is the first non-blank character after the = sign. If the line is empty after the =, a blank is used. The default is a colon. Note that a space will always follow the separator in the report.

NUMBER_OF_0s_TO_ROUND = If this number of consecutive 0s appears to the right of the decimal place, followed by nonzero values, round them to 0. The default is 3.

NUMBER_OF_9S_TO_ROUND = If this number of consecutive 9s appears to the right of the decimal place, round them up to 1. The default is 3.

REPORT_COLUMN_WIDTH = Default width in characters of report columns. This value may be increased, depending on the data displayed in the column. Must be between 8 and 32, inclusive. The default is 16.

Oracle Utilities Billing Component Settings:

AUTOBILL = This line can be repeated. Each line should contain the name of one machine that is allowed to run Automatic Billing. The name of each machine is listed in the About box of any Oracle Utilities application run from the machine.

AUTO_BILL_MAX_COUNT = Number. During each run of Automatic Billing and Approval Required, Oracle Utilities Billing Component finds all eligible accounts and tries to bill each. This parameter can be used to limit the size of this list. It should be used when the available memory is limited. A typical number for a medium size server is 100000; for a desktop machine it might be 10000. Try setting a smaller number if Automatic Billing is slow.

AUTO_BILL_NO_BILL_CYCLE = 1: If this line is present, checking for a Billing Cycle and Billing Cycle Date when processing billing for accounts is disabled. Accounts must still have an active Account History record in place, but Billing Cycle information is not needed (and ignored if present). This option should **ONLY** be used in conjunction with on-demand Bill Determinant calculations initiated via integration between Oracle Utilities Meter Data Management and a customer information system such as Oracle Utilities Customer Care and Billing. See **Appendix B: Integrating Oracle Utilities Meter Data Manager with a Customer Information System** for more information.

CHECK_FINAL_BILL: If this parameter is present (no value needed), when approving a Final Bill for an account or customer Oracle Utilities Billing Component prompts the user with the question: "Are you sure you want to Final Bill? (Yes/No)". If they answer "Yes" a final bill will be created. If they answer "NO" the bill will not be approved.

Note: This message will not display when using the "Automatically Save" approval options.

This option works with the client/server version of Oracle Utilities Billing Component **only**.

CISFILENAME = The default output CIS file name, as displayed in the Oracle Utilities Billing Component dialogs. It is also the CIS output file name used by AUTOBILL, if not overridden on the command line. If not set, the default file is "LODESTAR.CIS" in the USER directory.

CISFORMAT = The input file name containing the CIS record format text. If a file name is supplied with no path, the file must be in the USER directory. The default is CISFORMAT.TXT in the CFG directory.

DELAYED_CIS_FILE_OPEN = Normally the CIS output file is opened in append mode during the validation done when a bill is first approved. This detects any file problems quickly, however, it leaves the file open while the database records are saved. If you want the file to be open for as short a period of time as possible, put this keyword in the configuration file (no value needed). The CIS file will then be opened just before the database changes are committed.

DIRECT_WRITE_CIS: If present and if saves are enabled, the system writes the CIS records directly to the CIS output file. This bypasses any user approval. If an error occurs after CIS records are saved, they will still be in the CIS file. This should be used only when the records are very large or numerous and there is not enough memory to store them until approved.

Note: If this is used, all executed SAVE TO CIS statements must include a section name, and all field identifiers in the section must be set in the rate schedule - there is no default fill-in of account and bill date information.

Note: When this parameter is used in conjunction with the `RELEASE_SUMMARY_INFO` parameter, CIS records are written only if the account is successfully processed.

IGNORE_CIS_FILE: If this parameter is present (no value needed), CIS file errors are ignored when approving bills. This allows approvals to proceed whether the CIS file is opened or not.

INVOICENAME = the default INVOICENAME (from the Invoice Number table) for invoices generated by Oracle Utilities Billing Component. See **Invoice Numbering** in **Chapter 4: Billing Rules and Definitions** in the *Oracle Utilities Billing Component Installation and Configuration Guide, Volume 1* for more information about Invoice Numbering.

LS_HISTORICAL_DATA_SUBSTITUTION: Overrides the default behavior of missing historical data when using the `COMPSUM`, `MAXNRANGE`, `MAXRANGE`, `MINRANGE`, `MAXSEASON`, `MINSEASON`, `SUMSEASON`, or `AVGSEASON` Rules Language functions. Can be set to one of the following three values. If set to 0, the functions return a zero value when the specified historical data is missing. If set to 1, the functions return an error message when the specified historical data is missing. If set to 2, the functions use the oldest available value when the specified historical data is missing. If this parameter is set to any other value or not specified, then default behavior is used.

NO_BH_AUDIT_TRAIL: If this keyword is present, writes to the Bill History Edit and Bill History Edit Value tables are disabled, improving performance of Oracle Utilities Billing Component. This parameter should only be included if the auditing functionality of the Bill History Edit and Bill History Edit Value tables is not used.

REBILLREASONREQUIRED: If this keyword is present (no value needed) and there is a `REBILLREASON` Table in the database, a Rebill Reason Code must be selected to run **Rebill Only** or **Cancel/Rebill** in Bill Correction.

RELEASE_SUMMARY_INFO: If this keyword is present, the Rules Language engine will release memory after each account is processed when processing multiple accounts. This setting is recommended when processing large numbers of accounts using Autobill.

UOM_NO_INTD_DATE = This line can be repeated. Each line should contain one UOM code. If a code appears here and the UOM is the code in a `CHANNELHISTORY` record that is required for billing, the interval data will be validated as present, but will not be used to compute the bill period dates. If this parameter is not set, the interval data cuts will be used to compute the bill period dates.

USE_WORK_Q_FOR_ACCOUNT_NOTES = 1: If this line is present, account notes generated by Oracle Utilities Billing Component are stored in the Work Queue Open Items table instead of the Account Note table. The Account Notes tab of the Account-Centric View is not displayed if this list is present.

WRITE_BILLHISTORY_EDIT_UPON_INITIAL_BILL_APPROVAL = 1: If this line is present, records are written to the Bill History Edit and Bill History Edit Value tables upon the initial approval of a bill. This can help when reviewing previous changes and/or edits to Bill History records.

Data Manager Settings:

BROWSERENTITIES = If this keyword is present, only database entities are displayed in the database browser when the tables are first displayed (all tables are displayed on refresh).

BROWSTABLE = tablename: This line may be repeated with different table names, one name per line. If the `BROWSERENTITIES` keyword is not present, only the database tables with these names are displayed in the database browser when the tables are first displayed (all tables are displayed on refresh).

CASE_INSENSITIVE = 1: If this line is present, search in the record select dialog will be case insensitive. Default is case sensitive search.

DATABASE_MONITOR = 1: If this line is present, then database monitoring (used with early versions of the Oracle Utilities Transaction Management) is turned ON.

DM_BROWSER_MAX_RECORDS = the maximum number of records per table displayed in the Data Manager Browser. Records are sorted by their identity. For example, if this parameter is set to 50, and a user selects a table with 100 records, only the first 50 would be displayed in the browser. This parameter should be used when tables contain extremely large volumes of records. The default value is 0 (zero), which means that all records are displayed.

INTD_OLD_RECORDER_SELECT: The manner in which recorders are selected when browsing interval data using Data Manager has been changed to display only those recorders with interval data. If this keyword is present, Data Manager uses the previous method for selecting recorders, which lists all recorders in the Recorder table. Using this keyword improves performance when accessing large interval databases.

KEEP_UIDTOUSCHEDULE: If this keyword is present, the TOUSCHEDULE UID will not change when updates are made to a TOUSCHEDULE.

LOGTABLE: If this keyword is present, log records from importing customer/account data (using either Data Manager or PLIMPORT.EXE) are placed in the ERROR table in the Oracle Utilities Data Repository instead of into log files. Note that the ERROR table **must** be in the database for this option to function properly.

NO_GRAPH_LIMIT_MESSAGE: If this keyword is present, the Trial Calculation dialog box indicating that there are already 12 interval data graphs displayed will not appear. This allows trial calculations with more than 12 interval data graphs to be run without the need of closing the dialog.

Note: If you include this parameter, only the first 12 interval data graphs will have labels.

RSLOCK: If this keyword is present, Data Manager will lock rate forms while editing text.

STOPTIME_UPDATE = tablename: This line may be repeated with different table names, one name per line. Whenever a new record is inserted in a table specified with the STOPTIME_UPDATE keyword, the user will be prompted to update the Stop Time of a previous record. For example, if a record in the Channel History table for a given channel has a NULL Stop Time (and Channel History has been specified with the STOPTIME_UPDATE keyword), if a new record is inserted into the Channel History table for the same channel, the user will be prompted to update the Stop Time on the previous record.

Language Settings:

ALLOWINTDREASSIGN = Anything. The default is to generate an error if an interval data identifier is assigned any value that is not an interval data handle, or 0 (usually caused by a mistake in coding a rate schedule or its riders). Setting this to anything overrides this error check and allows any value to be assigned to an interval data identifier, possibly changing it to a temporary identifier.

ALLOWREVENUEIDREUSE = 1 or 2. The default is to generate a compile error if a revenue identifier is an INTO or TOTAL identifier in more than one BLOCK or ALL statement. If this parameter is set, this check is skipped: if the value is 1, a warning message is generated on validation; if it is 2, no warning messages are generated for these errors. If a revenue identifier is used more than once, results may be incorrect, because the information from one statement may be overwritten by that from another.

If this parameter is on, identifiers are not checked against the IDENTIFIER column in the BILDETERMINANT table. If this parameter is on, it will not be an error to assign a value to a revenue identifier used in a BLOCK or ALL statement, even though that may lead to missing information.

Note: These last two situations will be reported as Warnings when you validate a rate form, but will not interfere with saving or running it.

CACHE_COMPILED_RATE = 1 If present, the application caches compiled rate schedules. This is specifically useful when executing rate schedules via the Oracle Utilities Adapter where the same rate is run multiple times.

Note: When using this setting and running multiple different rate schedules, the Shared Symbol table will contain identifiers from all the rate schedules executed. In this case, identifiers used in the first rate schedule may appear in the output reports for subsequent rate schedules unless the identifiers are explicitly cleared via the CLEAR or REMOVE statements.

DONOT_CHECK_SAVETOTABLE_BH_AND_BHV: If this keyword is present, the Rules Language editor does not display an error dialog when using the SAVE TO TABLE statement when saving records to the Bill History and Bill History Value tables.

DO_NOT_UPDATE_AUDIT_TRAIL_FOR_RS_SAVES: If this key word is present, changes to the Bill History record via Rules Language are not recorded in Bill History Edit and Bill History Edit Value tables as a separate entry. Changes made via Rules Language are recorded only once when the bill is approved. Without this parameter, each time a Bill History record is changed within a rate schedule, a new Bill History Edit record is created, along with corresponding Bill History Edit Value records.

ERROR_ON_MISSING_LIST_ID: If this keyword is present, it is an error if an identifier, when used as a clause identifier for a list, is used prior to the identifier being assigned in the rate schedule.

INTDLOADLIST_STOP_ON_LIST_ERROR = 1 If present, it is an error if a list used by the INTDLOADLIST function does not exist.

OVERRIDE_NULL_VAL_IS_ZERO = Anything. In previous versions, if an override record was missing or its VAL or STRVAL columns were NULL, OVERRIDE[key].VAL would be set to 0.0, OVERRIDE[key].STRVAL would be set to "", and the HASVALUE function would indicate that they had a value. As of release 2.10.001, these will be set to NULL and HASVALUE will indicate that they do not have a value. To retain the previous assignment of 0.0 or "", set this configuration parameter to anything. See **Override Identifiers** in **Chapter 4: Identifiers, Constants, and Expressions**, in the *Oracle Utilities Rules Language User's Guide*.

PARSER_LIMIT = Number. The internal parsing limit of the rate schedule compiler. Default value is 200.

REMOVE_EMPTY_COLUMNS = Anything. If this is present, a column in a billing report that has no values will not be displayed. If this is not present, the column header will appear with nothing below it.

RETURN_UID_FOR_UID_TARGETED_LISTS: This enables the FOR EACH IN LIST Rules Language statement to return a UID if the list targets a UID column (as was the default behavior in versions prior to 2.10). The default behavior now is to return the identity (comma separated).

Note: Including both RETURN_UID_FOR_UID_TARGETED_LISTS and TARGET_IDS_FOR_UID_REF_LISTS in the same configuration file will result in an error.

SAVE_TO_TABLE_ADDFIRST = 1 If present, the SAVE TO TABLE statement will first try to ADD the record, and if that fails, then it tries to UPDATE the record.

TARGET_IDS_FOR_UID_REF_LISTS List queries targeting UID reference columns by default return identity columns from their referenced table. Setting this parameter returns the identity from the target table.

Note: Including both TARGET_IDS_FOR_UID_REF_LISTS and RETURN_UID_FOR_UID_TARGETED_LISTS at the same configuration file will result in an error.

USE_DOMDOCADDAPI_IN_LOOP = 1 Include this keyword and value when using the DOMDOCADDAPI XML Rules Language function inside a FOR EACH in LIST statement. Using the DOMDOCADDAPI XML Rules Language function inside a FOR EACH in LIST statement without this parameter can result in errors.

USE_DST_SCALING: By default, the INTDSCALE Rules Language function assumes a 24-hour day when scaling interval data down from daily intervals to smaller interval sizes. If this parameter is present, the function scales interval data based on the number of hours in the day (for example, 23 hours for the Spring DST day and 25 hours for the Fall DST day).

USE_GLOBAL_LIST_CACHE = 1 If present, list caching at the application level is enabled. By default lists are cached only within a rate. This is specifically useful when executing rate schedules via the Oracle Utilities Adapter where the same rate is run multiple times.

Performance Settings:

Customer Revenue = 6000: maximum number of customer accounts that can be processed in one analysis (Oracle Utilities Rate Management only.)

Customer Impact = 6000: maximum number of customer accounts that can be processed in one analysis (Oracle Utilities Rate Management only.)

INTDGETRCDRS = Value must be the number of recorders to display at a time. The smaller the number, the faster the first one appears; the larger the number, the sooner all are displayed. Minimum value is 2, maximum is 50. Default is 10.

INTERVAL_DATA_CACHE_BUFFER_SIZE Sometimes when loading large amounts of interval data, it is possible that memory can get fragmented and eventually run out. Setting this parameter helps to prevent memory fragmentation problems caused by loading interval data. By default, the size is 1048576 bytes (1MB). Can be set based on the size of the interval data cuts being loaded.

MAX_RECORDERS_PER_USERID = Maximum number of recorders authorized for any user. The default is 256; minimum value is 2.

MAXTABLECT = Value must be larger than the number of tables in the Customer Database. Defaults to 256 (which is also the minimum).

NODEFAULTUSER = If defined as anything, turns off getting a default user's value for the interval data source. If not set, the application attempts to load the DEFAULTUSER's options at logon and after every save of the user's options.

RATE_LANGUAGE_SYMBOLS = Estimated number of symbols in largest expanded rate schedule, plus number of REPORT statements executed, plus number of SAVE TO CIS statements executed, plus number of SAVE TO TABLE statements executed. This is used to initially allocate a table to hold all identifiers. A large value helps prevent thrashing. Default is 256. You can select Display Error Window after running a bill calculation to show you a setting for all bill calculations to that point.

Testing Settings:

AUTO_BILL_LOG_PROGRESS = 1: turns on logging the progress of Automatic Billing.

CISBILLOPTID = string. The usual, global ID used to store the CIS Billing Options is !CISBILLINGOPTIONS. If you need to set different CIS Billing Options for different users (e.g., for testing), you can change the ID used to store and retrieve these options on a user by user basis by adding this option to the configuration file. To make sure the ID is not an actual USER ID, use ! as the first character. This parameter does not apply when using the web-enabled version of Oracle Utilities Billing Component.

DEBUGPARM = 1: turns on debug mode.

DEFAULTUSER = Sets the userid used to get a default value for the interval data source. It is case sensitive. If not supplied, the default user id is "DEFAULTUSER".

GET_FULL_RS = 1: turns off expanding each INCLUDE statement only once when compiling a rate schedule. This will revert to pre-2.03.041 compiling of Rate Schedules.

INTD_TEST_SECURITY = If defined as anything, enables the LODEMAP security feature in Data Manager and Oracle Utilities Billing Component. This limits the displayed recorders to those that are in the Oracle Utilities Data Repository (RECORDER and ACCTOVERRIDEHIST tables) and are visible to the user.

MEM_USAGE_DISPLAY_RLP: If this key word is present, a memory snapshot that displays memory usage of the Data Manager application is included in any Rules Language code profiles. See **Rules Language Profiling** on page 14-2 for more information about Rules Language code profiles.

NOCATCH = Define as anything to turn off catching fatal errors and display Windows error message.

NO_TEMP_SAVE = If defined as anything turns off verifying saves to the database during a run. It should not be used if, in a bill calculation, you retrieve records that were previously saved during the same calculation. It is overridden (off) if one of the automatic commit modes is selected in Trial Bill, "Automatic Approve" is checked in Bill Correction or Approval Required, or you are using Automatic Billing or RUNRS.EXE. Otherwise, if this is not on then records are temporarily saved to the database to verify that their values are correct.

Report Testing = If defined (as anything), turns off the ": ##" in each window name. This allows for easier testing.

EXTENDEDSTATUSES.CFG.XML Configuration File

The EXTENDEDSTATUSES.CFG.XML file can be used to define custom extended status codes for interval data readings and how these custom codes are displayed in the Oracle Utilities Energy Information Platform user interface. The EXTENDEDSTATUSES.CFG.XML file must be installed in the **C:\LODESTAR\CFG** directory.

A sample of this file is provided in the **C:\LODESTAR\CFG\Examples\CFG** directory.

Extended status codes are defined in meter readings as integer values starting at 1. In the EXTENDEDSTATUSES.CFG.XML file, status codes are defined by the order in which the status codes exist, from 0 through 15. These locations correspond to integer values. For example, using the example below, the channel status code for a reading with a CHNSTATUS (channel status code) of “128” would be displayed as “PY” (code), “Parity Error (description).

EXTENDEDSTATUSES.CFG.XML Example

```
<EXTENDEDSTATUSES>
  <STATUS_SET TYPE="CHANNEL">
    <STATUS BIT="0" CODE="NA" DESCRIPTION="Not Used"/>
    <STATUS BIT="1" CODE="AD" DESCRIPTION="Added Interval (Data Correction)"/>
    <STATUS BIT="2" CODE="RE" DESCRIPTION="Replaced Interval (Data Correction)"/>
    <STATUS BIT="3" CODE="ES" DESCRIPTION="Estimated Interval (Data Correction)"/>
    <STATUS BIT="4" CODE="OV" DESCRIPTION="Pulse Overflow"/>
    <STATUS BIT="5" CODE="HL" DESCRIPTION="Data Out Of Limits"/>
    <STATUS BIT="6" CODE="XC" DESCRIPTION="Excluded Data"/>
    <STATUS BIT="7" CODE="PY" DESCRIPTION="Parity Error"/>
    <STATUS BIT="8" CODE="TY" DESCRIPTION="Energy Type (Register Changed)"/>
    <STATUS BIT="9" CODE="LR" DESCRIPTION="Alarm/Error"/>
    <STATUS BIT="10" CODE="DI" DESCRIPTION="Harmonic Distortion"/>
    <STATUS BIT="11" CODE="IN" DESCRIPTION="Point-to-Point Linear Interpolation"/>
    <STATUS BIT="12" CODE="CK" DESCRIPTION="Check Estimation"/>
    <STATUS BIT="13" CODE="NA" DESCRIPTION="Not Used"/>
    <STATUS BIT="14" CODE="NA" DESCRIPTION="Not Used"/>
    <STATUS BIT="15" CODE="NA" DESCRIPTION="Not Used"/>
  </STATUS_SET>
  <STATUS_SET TYPE="INTERVAL">
    <STATUS BIT="0" CODE="PO" DESCRIPTION="Power Outage"/>
    <STATUS BIT="1" CODE="SI" DESCRIPTION="Short Interval (False For Mag Tape)"/>
    <STATUS BIT="2" CODE="LI" DESCRIPTION="Long Interval (Missing For Mag Tape)"/>
    <STATUS BIT="3" CODE="CR" DESCRIPTION="CRC Checksum Error"/>
    <STATUS BIT="4" CODE="RA" DESCRIPTION="RAM Checksum Error"/>
    <STATUS BIT="5" CODE="RO" DESCRIPTION="ROM Checksum Error"/>
    <STATUS BIT="6" CODE="LA" DESCRIPTION="Data Missing"/>
    <STATUS BIT="7" CODE="CL" DESCRIPTION="Hardware Clock Error"/>
    <STATUS BIT="8" CODE="BR" DESCRIPTION="Memory Reset"/>
    <STATUS BIT="9" CODE="WT" DESCRIPTION="Watchdog Timeout"/>
    <STATUS BIT="10" CODE="TR" DESCRIPTION="Time Reset"/>
    <STATUS BIT="11" CODE="TM" DESCRIPTION="Test Mode"/>
    <STATUS BIT="12" CODE="LC" DESCRIPTION="Load Control"/>
    <STATUS BIT="13" CODE="NA" DESCRIPTION="Not Used"/>
    <STATUS BIT="14" CODE="NA" DESCRIPTION="Not Used"/>
    <STATUS BIT="15" CODE="NA" DESCRIPTION="Not Used"/>
  </STATUS_SET>
</EXTENDEDSTATUSES>
```

EXTENDEDSTATUSES.CFG.XML Element Descriptions

Each of the data elements used by the EXTENDEDSTATUSES.CFG.XML file are described below.

EXTENDEDSTATUSES: Root element for the file. Can contain one or more STATUS_SET elements.

STATUS_SET: Element that defines a set of extended status codes.

Attributes:

TYPE: Defines the type of extended status code defined in this STATUS_SET element. Valid values are “CHANNEL” (channel-level status codes) and “INTERVAL” (interval-level status codes)

Elements:

STATUS: Defines a specific extended status code of the type defined in the parent STATUS_SET element

Attributes:

BIT: Defines the order in which the status codes exist, from 0 to 15. Each BIT corresponds to an integer value, starting at 1 through 32768, where each value is equal to twice the preceding value (1, 2, 4, 8, 16, 32, etc.). Integer values greater than 32768 (BIT 15) represent the sum of smaller integer values. The integer values that correspond to each bit are as follows:

Bit	Integer Value
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096
13	8192
14	16384
15	32768

Multiple extended status codes can be specified in a reading by an integer that is equal to the sum of the integer values that correspond to the individual status codes. For example, to specify interval status codes “PO” (1) and “SI” (2), you would set this equal to “3”. To

specify interval status codes “PO” (1), “SI” (2), and “CR” (8), you would set this equal to “11”.

CODE: A code that designates the extended status code. The code (and description) are displayed in the user interface on the Edit Values and Edit Header screens.

DESCRIPTION: A description of the extended status code. The description (and code) are displayed in the user interface on the Edit Values and Edit Header screens.

IDV.CFG.XML

The IDV.CFG.XML file defines the behavior of specific features of the Interval Data Viewer and Interval Data Manager. If used, the IDV.CFG.XML file should be included in the **C:\LODESTAR\CFG** directory on the web server. If this file is not present, the default settings (as shown in the example below) are used.

A sample of this file is provided in the **C:\LODESTAR\CFG\Examples\CFG** directory.

IDV.CFG.XML Example

```
<IntervalData>
  <AnalysisGraph>
    <InteractiveValues default="No" />
    <Y_Axis MinMaxPercent="3" />
    <ScaleSPI default="IDVSRMAX" />
  </AnalysisGraph>
  <EditValues>
    <Decimal Digits="5" />
  </EditValues>
</IntervalData>
```

IDV.CFG.XML Element Descriptions

Each of the data elements used by the IDV.CFG.XML file are described below.

IntervalData: The root element of the IDV.CFG.XML file.

Elements:

AnalysisGraph: Element that defines the behavior of specific features of the Interval Data Summary screen.

Elements:

InteractiveValues: Element that specifies whether or not interval values are interactively displayed in the lower right corner of the graph pane on the Interval Data Summary screen.

Attributes:

default: Specifies whether or not interval values are interactively displayed in the lower right corner of the graph pane on the Interval Data Summary screen. Valid values are “Yes” and “No” (default).

Y_Axis: Element that specifies the manner in which the lowest graphed value is depicted on the graph pane.

Attributes:

MinMaxPercent: The percentage of the height of the graph, used to define the amount of space between the bottom of the graph and the point on the y-axis where the lowest graphed value is displayed. For example, if set to 5, the lowest graphed value would be displayed at a point on the Y-axis that is a distance above the bottom of the graph equal to 5 percent of the total height of the graph. The default value is “3.”

ScaleSPI: Element that specifies the default manner in which interval data of different interval lengths is scaled when graphed together.

Attributes:

default: Specifies how interval data of different interval lengths is scaled when graphed together. Can be either scaled based on the Shortest Intervals (“IDVSRMIN”), or Longest Intervals (“IDVSRMAX”). The default setting is “IDVSRMAX.”

EditValues: Element that defines the manner in which interval values are displayed on the Interval Data Manager Edit Values screen.

Elements:

Decimal: Element that defines the number of decimal places displayed for interval values in the Cut Values list box on the Edit Values screen.

Attributes:

Digits: The number of decimal places displayed for interval values in the Cut Values list box on the Edit Values screen. The default value is “5.”

INTDCONFIG.CFG.XML

The INTDCONFIG.CFG.XML file defines various interval data behavior properties, including how interval time stamps are displayed, the default behavior when importing and exporting interval data, and how the default Unit-of-Measure list used by Oracle Utilities applications can be overridden and/or extended. If used, the INTDCONFIG.CFG.XML file should be included in the **C:\LODESTAR\CFG** directory on the web server. If this file is not present, the default settings (as shown in the example below) are used.

A sample of this file is provided in the **C:\LODESTAR\CFG\Examples\CFG** directory.

INTDCONFIG.CFG.XML Example

```
<Rules>
  <IntervalTimeStamp Show = "IntervalStart" />
  <INTDBehavior>
    <Import OverCount = "Reject"
      UnderCount= "BE"
      Set_Invalid_DST_Participant = "Reject"
      DST_Participant = ""
      Set_Missing_DST_Participant = "Y"
      Set_DST_A_Records = "Y"/>
    <Export Default_Validate_Record_Flag = "Flag" />
    <ALLOW_ANY_RECORDERID_CHAR/>
  </INTDBehavior>
</SESSION/>
<UOMSource>
  <DATASOURCE>
    <NAME>UOM_DS</NAME>
    <CONNECTSTRING>Data Source=TS92;User
ID=ls105d;Password=password;LSProvider=ODP</CONNECTSTRING>
    <QUALIFIER>pwrline</QUALIFIER>
  </DATASOURCE>
</UOMSource>
</Rules>
```

INTDCONFIG.CFG.XML Element Descriptions

Each of the data elements used by the INTDCONFIG.CFG.XML file are described below.

Rules: The root element of the INTDCONFIG.CFG.XML file.

Elements:

IntervalTimeStamp: The element that defines how the interval time stamp is displayed on the Interval Data Summary screen accessed using Interval Data Viewer. If this element is not present, the interval time stamp is displayed as the interval Start Time.

Attributes:

Show: Defines how the interval time stamp is displayed. Valid values include:

IntervalStop: Displays interval time stamps as the end of the interval. For example, for a 15 minute interval that spans from 00:00:00 to 00:14:59, this option would display 00:14:59.

IntervalStart (default): Displays interval time stamps as the start of the interval. For example, for a 15 minute interval that spans from 00:00:00 to 00:14:59, this option would display 00:00:00.

NextIntervalStart: Displays interval time stamps as the start of the next interval. For example, for a 15 minute interval that spans from 00:00:00 to 00:14:59, this option would display 00:15:00.

INTDBehavior: The root element that defines the section in the INTDCONFIG.CFG.XML file that contains elements and element attributes that describe the desired behavior of interval data handling functions.

Elements:

Import: Defines the default behavior when importing interval data.

Attributes:

OverCount: Controls the behavior of import functions when an incoming cut has too many recorded intervals for the time span defined by the cut start time and the cut stop time and the DST_Participant flag.

Valid Values:

REJECT: (Default) Directs the function to produce an error and disallow the import of any incoming/uploaded when cut has too many recorded intervals (Expected < Recorded).

BE: Behaves exactly like REJECT, except in the case where a cut that crosses the beginning of a DST period, and has its recorded interval count equal to the expected intervals count without the hour adjusted for DST.

ASIS: Allow the storage of any incoming cuts with a number of expected intervals less than the number of recorded intervals. The stop times of these cuts will remain unchanged.

UnderCount: Controls the behavior of import functions when an incoming cut has too few recorded intervals for the time span defined by the cut start time and the cut stop time and the DST Participant flag.

Valid Values:

REJECT: Directs the function to produce an error and disallow the import of any incoming/uploaded cut when it has too few recorded intervals (Expected > Recorded).

BE: (Default) Directs the function to behave similar to the default behavior observed in previous installations and versions of Oracle Utilities products. This setting works exactly like FILL in most cases. When a cut being processed spans the end of the DST period, usually the fall DST hour, if viewed directly from the raw flat file, the cut will appear to be filled using DST calculations, yet if the cut is saved to any binary data source (RDB or BTE), it will only be filled with missings as if the DST_Participant flag is set to "N". For example, if a cut is supplied that crosses the Fall DST change and has only 92 intervals (23 hours), the system will fill to only 96 not 100. If 100 intervals are supplied, then 100 will be stored. Also, if 97 - 99 intervals are supplied and the DST = "Y", then the system will Reject the incoming cut.

FILL: Fills all incoming cuts with missings (value = 0, Status = "9") to the stop time of the cut using DST adjustments if appropriate (DST_Participant = "Y").

ASIS: Allows the storage of any incoming cuts with a number of expected intervals greater than the number of recorded intervals. The stop times of these cuts will remain unchanged.

Set_Invalid_DST_Participant: Controls the behavior of the import and loading of cuts when the DST_Participant flag in the source cut is not an A, N, or Y.

Valid Values:

Y: Sets the DST_Participant flag of an incoming cut with an invalid DST_Participant flag to Y.

N: Sets the DST_Participant flag of an incoming cut with an invalid DST_Participant flag to N.

Reject: Does not allow the cut to be imported (Default).

DST_Participant: Controls any Daylight Saving Time conversions that will be applied to the cut as it is imported into the Oracle Utilities system.

Valid Values:

Y: Converts all incoming cuts to follow the default DST rules set in LSCalendar.xml or uses the incoming cut's Time Zone Standard Name (TZSN) if available. This setting changes all incoming cuts with DST_Participant flags set to "A" or "N" to "Y" and converts the cuts using the DST conversion algorithm for that TZSN.

N: Converts all incoming cuts not to follow the DST rules associated with them via the TZSN or by using the default set in LSCalendar.xml. This setting changes all incoming cuts with DST_Participant flags set to "A" or "Y" to "N".

ASIS: (Default) Allows cuts to be entered into the system without using any conversions. The DST_Participant flags remain as they are supplied. This can result in databases that have mixed DST_Participant cuts, and is not recommended if the data source will be used with Oracle Utilities Load Analysis.

Set_Missing_DST_Participant: Controls the default value assigned to the DST_Participant Flag when the incoming cut does not contain the flag.

Valid Values:

Y: (Default) Sets the DST_Participant flag to "Y".

N: Sets the DST_Participant flag to "N".

Set_DST_A_Records: Sets rules on how to import cuts with the Participant flag set to "A".

Valid Values:

Y: (Default) Convert the cut from a properly formatted "A" record to one that follows DST rules, provided the TZSN is also supplied. If the TZSN is not supplied, the function uses the default TZSN. The cut will be transformed into the original DST = "Y". When using Time Zone Conversions in Data Manager or the INTDIMPEXE command line program, the program will only import cuts in the LSE, XML, CSV file formats that originally have the requested DST_Participant flag. All others produce an error. For Example, if an incoming file has a mixture of DST_Participant cuts, and the Input/Meter Time Zone is EDTA, then only cuts in the file with DST_Participant flags of A will be imported. The cuts will be converted to DST = "Y" in this case.

N: Results in the same processing as the "Y" setting, followed by a conversion of start and/or stop times by one hour to create a non-DST cut.

Reject: Results in the cut being rejected and produces an error.

Asis: Imports as Data Manager has always done it.

Export: Defines the default behavior interval data assumes when exporting interval data to flat file formats from IDM only:

Attributes:

Default_Validate_Record_Flag: Controls the default behavior of how the system sets the Validate_Record_Flag in the LSE format.

Valid Values:

Y: Sets all exported cut attributes Validate_Record_Flags to "Y" indicating that the record should be validated after importing.

N: Sets the Validate_Record_Flag on all exported cuts to "N" indicating that the cut needs no further validation when imported.

Flag: (Default) Uses the cuts' Internal_Validation and Merge flag to set the Validate_Record_Flag. If either are set to "Y", then the Validate_Record_Flag is set to "N", otherwise it is set to "Y".

ALLOW_ANY_RECORDERID_CHAR: If present, Recorder ID character validation upon import is disabled, allowing any character to be used as part of a Recorder ID when importing interval data.

SESSION: Element containing interval data behavior settings. To use default parameters, include empty session element (<SESSION/>).

Elements:

QualityCode: Ignore interval data values whose status code is worse than the specified QualityCode. Valid values for this element are [A-Z] | [0-9], space, or no value. If no value is present then a status code of space is used. If the QualityCode element is not present, the default used is a value of 8. If the status code of an interval is worse than the character specified in the QualityCode code element, then that interval will be returned as a missing interval (Value of 0, Status code '9').

DayStart: Defines the start of the day. Valid values are 0 - 11, where 0 is for Midnight, 1 is for 1 A.M., 2 is for 2 A.M., up to 11 for 11 A.M. If DayStart element is not present, the default value is 0 (Midnight).

INTD_NEW_JOIN: Determines the behavior when merging interval data cuts.

Attributes:

INTDMERGESELECT: *Optional.* Specifies the criteria for merging overlapping interval cuts (i.e., a rule how to merge two cuts into one).

- **RECENT:** Pick the interval from the cut that starts later.
- **AVERAGE:** Pick the time-weighted average of the overlapping intervals. The status code of the averaged interval will be the worse of the overlapping intervals.
- **BEST:** Pick the interval from the cut with a better status code. If the intervals have the same status code, the values are averaged.
- **MAXIMUM:** Pick the interval from the cut with a larger value.
- **TIMESTAMP:** Pick the interval from the cut with the later Timestamp.

CHECKVALID: *Optional.* Specifies whether or not to check cuts' merge validation flags before merging. Valid values are "Y" (Yes) and "N" (No). The default is to NOT check validation flags.

CHECK_UOM_MATCHING: If this element is present, the interval data component checks UOM Compatibility when merging cuts across a specified date range; by default the component does not check UOM compatibility.

Attributes:

CHECK_COMPATIBILITY: Specifies whether or not to check the UOM's compatibility (matching) using Oracle Utilities Load Analysis. Valid values are "Y" (Yes) and "N" (No). The default is to NOT check compatibility.

DO_NOT_USE_UOM_FOR_SCALING: If this element exists, interval data is totalized when scaling or aggregating, regardless of the Totalize Method flag specified in the Unit-of-Measure table.

JOIN_ROUND_START_TIME: If present, when two cuts within the same cut series are merged and partial intervals are present in both cuts, the start is rounded to the nearest boundary so that the number of intervals in the merged cut can be calculated correctly.

UOMSource: Element that defines an alternative source of Unit-of-Measure codes (UOMs) that can be used by Oracle Utilities applications. This is used when adding custom UOMs to the Oracle Utilities Data Repository for use with Interval Data Manager (IDM).

Elements:

UOMList: Element that defines a list of custom UOMs that will extend the default list of valid UOMs available with Interval Data Manager (IDM).

Elements:

- **LSUOM:** Element that defines a single UOM.

Attributes:

CODE: unique code (up to 64 characters) identifying the unit of measure

NAME: complete name or other description for the UOM (up to 64 characters). This name helps users recognize the code in displays and reports.

UNIT: The physical unit measured; for example, the physical unit for the UOMs KWH-IN and KWH-OUT is KWH.

AGGREGATE: The method the programs will use to aggregate data when converting interval data in this UOM to a lower frequency. For example, kW measured every 15 minutes should *not* be added together when aggregating to hourly data, but should have an average or a maximum taken. Can be either A (Average: average interval values to get aggregate) or T (Total: add values to get aggregate).

TOTALIZE: Specifies how the programs will treat values for intervals or determinants to arrive at a total when combining data from two or more recorder channels, or when creating bill frequency tables. Can be either A (Average: average interval values to get total) or T (Total: add values to get total).

DATASOURCE: Element that defines the data source where the Unit-of-Measure table used to extend the default list of valid UOMs available with Interval Data Manager (IDM) is located.

Elements:

- **NAME:** The name of the data source.
- **CONNECTSTRING:** The connect string for the Oracle Utilities Data Repository:

For Oracle databases:

```
<CONNECTSTRING>Data Source=<data_source>;User
ID=<user_id>;Password=<password>;LSProvider=ODP;</CONNECTSTRING>
```

For Microsoft SQL databases:

```
<CONNECTSTRING>Data Source=<address>;Initial
Catalog=<SQL_database>;User
Id=<user_id>;Password=<password>;MultipleActiveResultSets=True;LSProvid
er=MSSQL;</CONNECTSTRING>
```

where:

- <data_source> is the Oracle TNS Name for the data source, from the TNS_NAMES.ora file (typically located in the \\<machine>\oracle\network\admin directory)
- <user_id> is the user ID for the database connection
- <password> is the password for the supplied user ID.
- <address> is the IP address or Hostname of the MS SQL Server database server
- <SQL_database> is the name of the MS SQL Server database
- **QUALIFIER:** The qualifier or schema used for database object access (tables, views, etc.).

LOCALES.CFG.XML

The LOCALES.CFG.XML file specifies the different regional locales that are available to the web-enabled Customer Choice Suite.

The LOCALES.CFG.XML installed contains a number of pre-configured locales from North America and Western Europe. Only those locales included in the LOCALES.CFG.XML file as delivered by Oracle Utilities are supported.

The LOCALES.CFG.XML file is installed in the **C:\LODESTAR\WEB\CCS** directory on the web server. In addition, a copy of this file is also provided in the **C:\LODESTAR\CFG\Examples\CFG** directory on the web server.

LOCALES.CFG.XML Example

```
<LOCALES>
  <LOCALE><ID>0x0409</ID><NAME>English (United States)</NAME></LOCALE>
  <LOCALE><ID>0x0809</ID><NAME>English (United Kingdom)</NAME></LOCALE>
  <LOCALE><ID>0x0407</ID><NAME>German (Germany)</NAME></LOCALE>
  <LOCALE><ID>0x0410</ID><NAME>Italian (Italy)</NAME></LOCALE>
  <LOCALE><ID>0x0419</ID><NAME>Russian</NAME></LOCALE>
</LOCALES>
```

LOCALES.CFG.XML Element Descriptions

Each of the data elements used by the LOCALES.CFG.XML file are described below.

LOCALES: Root element that contains available locales.

LOCALE: Defines individual locale.

Elements:

ID: The identifier for the locale.

NAME: The name of the locale.

Important

This file **MUST** be edited to specify available locales prior to running Oracle Utilities applications.

Locale Information

Locale information comes from the **Table of Language Identifiers** from the **National Language Support** section of the **MSDN Library**. You can access this information using Microsoft Internet Explorer at the following URL:

http://msdn.microsoft.com/library/en-us/intl/nls_238z.asp

LSCALENDAR.CFG.XML

The LSCALENDAR.CFG.XML can be used to define Daylight Savings Time rules for regions outside the continental United States. If used, this configuration file must be present in the **C:\LODESTAR\CFG** directory. If not present, then a default configuration will be used.

A sample of this file is provided in the **C:\LODESTAR\CFG\Examples\CFG** directory.

LSCALENDAR.CFG.XML Schema

The XML schema for the configuration file is shown below:

```
<xsd:schema targetNamespace="www.lodestarcorp.com/LSCalendar"
xmlns:cal="www.lodestarcorp.com/LSCalendar" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xsd:element name="CALENDAR">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="DSTREGIONS">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="DSTREGION">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="PERIOD" maxOccurs="unbounded">
                      <xsd:complexType>
                        <xsd:attribute name="STARTTIME" type="xsd:dateTime" use="required"/>
                        <xsd:attribute name="STOPTIME" type="xsd:dateTime" use="required"/>
                        <xsd:attribute name="OFFSET" type="xsd:double" default="1.0"/>
                      </xsd:complexType>
                    </xsd:element>
                  </xsd:sequence>
                  <xsd:attribute name="NAME" type="xsd:string" use="required"/>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="TIMEZONES">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="TIMEZONE" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:attribute name="STDNAME" use="required">
              <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                  <xsd:maxLength value="32"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:attribute>
            <xsd:attribute name="UTCOffset" type="xsd:double" use="required"/>
            <xsd:attribute name="DSTREGION" type="xsd:string" use="optional"/>
            <xsd:attribute name="DSTNAME" use="optional">
              <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                  <xsd:maxLength value="32"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:attribute>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
      <xsd:attribute name="DEFAULT" type="xsd:string" use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:key name="TIMEZONE_STDNAME">
    <xsd:selector xpath="cal:TIMEZONE"/>
    <xsd:field xpath="@STDNAME"/>
  </xsd:key>
  <xsd:keyref name="TIMEZONE_DSTREGION" refer="cal:DSTREGION_NAME">
    <xsd:selector xpath="cal:TIMEZONE"/>
    <xsd:field xpath="@DSTREGION"/>
  </xsd:keyref>
  <xsd:keyref name="DEFAULT_TIMEZONE" refer="cal:TIMEZONE_STDNAME">
    <xsd:selector xpath="."/>
    <xsd:field xpath="@DEFAULT"/>
  </xsd:keyref>
</xsd:element>
</xsd:sequence>
```

```
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

LSCALENDAR.CFG.XML Example

```
<CALENDAR xmlns="www.lodestarcorp.com/LSCalendar">
  <DSTREGIONS>
    <DSTREGION NAME="US">
      <PERIOD STARTTIME="1995-04-02T02:00:00" STOPTIME="1995-10-29T01:00:00"/>
      <PERIOD STARTTIME="1996-04-07T02:00:00" STOPTIME="1996-10-27T01:00:00"/>
      <PERIOD STARTTIME="1997-04-06T02:00:00" STOPTIME="1997-10-26T01:00:00"/>
      <PERIOD STARTTIME="1998-04-05T02:00:00" STOPTIME="1998-10-25T01:00:00"/>
      <PERIOD STARTTIME="1999-04-04T02:00:00" STOPTIME="1999-10-31T01:00:00"/>
      <PERIOD STARTTIME="2000-04-02T02:00:00" STOPTIME="2000-10-29T01:00:00"/>
      <PERIOD STARTTIME="2001-04-01T02:00:00" STOPTIME="2001-10-28T01:00:00"/>
      <PERIOD STARTTIME="2002-04-07T02:00:00" STOPTIME="2002-10-27T01:00:00"/>
      <PERIOD STARTTIME="2003-04-06T02:00:00" STOPTIME="2003-10-26T01:00:00"/>
      <PERIOD STARTTIME="2004-04-04T02:00:00" STOPTIME="2004-10-31T01:00:00"/>
      <PERIOD STARTTIME="2005-04-03T02:00:00" STOPTIME="2005-10-30T01:00:00"/>
    </DSTREGION>
  </DSTREGIONS>
  <TIMEZONES DEFAULT="EST">
    <TIMEZONE STDNAME="EST" UTCOFFSET="-5" DSTREGION="US" DSTNAME="EDT"/>
    <TIMEZONE STDNAME="CST" UTCOFFSET="-6" DSTREGION="US" DSTNAME="CDT"/>
    <TIMEZONE STDNAME="MST" UTCOFFSET="-7" DSTREGION="US" DSTNAME="MDT"/>
    <TIMEZONE STDNAME="PST" UTCOFFSET="-8" DSTREGION="US" DSTNAME="PDT"/>
  </TIMEZONES>
</CALENDAR>
```

Note: The above example includes DST rules for the years 1995 through 2005. The default configuration includes DST rules for the years 1970 through 2050.

LSCALENDAR.CFG.XML Element Descriptions

Each of the data elements used by the LSCALENDAR.CFG.XML file are described below.

CALENDAR: Root element that contains available Regions and Time zones.

DSTREGIONS: Element that contains defined DST Regions.

DSTREGION: Defines individual region.

Attributes:

NAME: The Name of the DST Region. This must be the same as the DSTREGION attribute defined in one of the TIMEZONE elements.

Elements:

PERIOD: A single DST period, usually defined as a year.

Attributes:

STARTTIME: The **local standard time** at which DST starts in the DSTREGION's time zone.

STOPTIME: The **local standard time** at which DST ends in the DSTREGION's time zone.

TIMEZONES: Element that contains defined Time Zones.

Attributes:

DEFAULT: The default time zone for the system. **Required.**

TIMEZONE: Defines individual time zone.

Attributes:

STDNAME: The standard name of the time zone.

UTCOffset: *Optional.* The number of hours west of GMT.

DSTRegion: The DST Region in which the time zone resides. This is used by the NAME attribute of the DSTREGION element.

DSTName: *Optional.* The Daylight Saving Time name of the time zone.

Notes

The STARTTIME and STOPTIME attributes of the PERIOD element are specified in **local standard time**. For example, the current US rules dictate that DST begins at **2:00 AM standard time** in the spring and ends at **1:00 AM standard time** (which is typically described as 2:00 AM DST) in the fall.

LSDB.CFG.XML

The LSDB.CFG.XML file defines a log file used to capture all ODBC statements (queries) executed by Oracle Utilities applications. The LSDB.CFG.XML file must be located in the **C:\LODESTAR\CFG** directory.

NOTE: This file should only be used when troubleshooting performance issues or other problems, as creating the SQL tracing file will impact performance.

A sample of this file is provided in the **C:\LODESTAR\CFG\Examples\CFG** directory.

LSDB.CFG.XML Example

```
<LS_SQL_CONFIG>
  <SQL_TRACE ENABLE="YES" LOG_PARAMS="YES" FILENAME="U:\SQLLOG.LOG"
  APPEND="NO"/>
</LS_SQL_CONFIG>
```

LSDB.CFG.XML Element Descriptions

Each of the data elements used by the LSDB.CFG.XML file are described below.

LS_SQL_CONFIG: Root element for the file.

Elements:

SQL_TRACE: Element that defines behavior of SQL tracing.

Attributes:

ENABLE: Specifies ("YES" or "NO") whether SQL tracing is enabled or disabled. If not specified, the default is "NO."

LOG_PARAMS: Specifies ("YES" or "NO") whether bind parameters are logged. If not specified, the default is "NO."

FILENAME: Specifies the path and file name where the SQL tracing will be logged. If not supplied, the default is

C:\LODESTAR\Log\LSSQLTrace_<application_name>.log.

where:

<application_name> is the name of the application ("datamgr" for Data Manager, "plbx" for c/s Oracle Utilities Billing Component, etc.)

APPEND: Specifies ("YES" or "NO") if SQL tracing information is to be appended to the file. If not supplied (or "YES" is specified), the default behavior is to append the file. If "NO" is specified, a new files created each time the application is run.

LSLOGGER.CFG.XML

The LSLOGGER.CFG.XML file defines how log files are generated by Oracle Utilities applications. Each Oracle Utilities component can be configured to write messages to a specified log file or an Event Log. If used, the LSLOGGER.CFG.XML file must be installed in the **C:\LODESTAR\CFG** directory.

NOTE: This file should only be used when troubleshooting performance issues or other problems, as tracing and logging will impact performance.

A sample of this file is provided in the **C:\LODESTAR\CFG\Examples\CFG** directory.

LSLOGGER.CFG.XML File Structure

The file structure of the LSLOGGER.CFG.XML file is as follows:

```
<lslogger
  [minlevel = "INFO | WARN | ERROR | FATAL"]
  [lang = "ENU"]
  [runtime_change = "YES | NO"]
  [dump_on_se = "YES | NO"]
>
  [<file
    [path = "lodestar.log"]
    [overwrite = "YES | NO | 1 | 0"]
    [delimiter = "|"]
    [minlevel = "INFO | WARN | ERROR | FATAL"]
    [lang = "ENU"]
    [trace = "YES | NO | 1 | 0"]
    [mode = "RESTRICTED , UNICODE , PLAIN , DELIMITED,
    NOUSERID, FILEPERPROCESS"]
    [processes = "ProcessId1[;ProcessIdN]"]
    [minlevel = "INFO | VERBOSE"]
    [rotate = "[1-9]+[0-9]*[K|M|G]"]
    [textwidth = "..."]
    [<include
      compid = "..."]
    [minlevel = "INFO | WARN | ERROR | FATAL"]
    />]
    [<excludecompid = "..."/>]
  />]
  [<systemlog
    [servername = "..."]
    [minlevel = "INFO | WARN | ERROR | FATAL"]
    [lang = "ENU"]
    [trace = "YES | NO | 1 | 0"]
    [mode = "RESTRICTED , UNICODE , PLAIN , DELIMITED, NOUSERID"]
    [processes = "ProcessId1[;ProcessIdN]"]
    [minlevel = "INFO | VERBOSE"]
    [<include
      compid = "..."]
    [minlevel = "INFO | WARN | ERROR | FATAL"]
    />]
    [<excludecompid = "..."/>]
  />]
</lslogger>
```

LSLOGGER.CFG.XML Element Descriptions

Each of the data elements used by the LSLOGGER.CFG.XML file are described below.

lslogger: Root element for logging configuration file, containing one or more **file** and/or **systemlog** elements.

Attributes:

minlevel: Indicates the minimum message level for log messages. Valid values are INFO, WARN, ERROR, or FATAL. The default is INFO.

lang: XXX, where XXX is a three letter language abbreviation that specifies the language of output text.

runtime_change: Indicates if the logging framework configuration can be changed in runtime (within a process) (“YES” or “NO”). The default is “YES”.

dump_on_se: Indicates if the logging framework should generate a minidump file when unhandled exceptions occur and when generation of a dump file is still possible. This dump file is created in the ..\Lodestar\log directory. The generated dump file can be analyzed in any debugger that can work with dump files.

Elements:

file: Element that defines how log messages are written to a log file.

Attributes:

path: Path and file name for output log file.

overwrite: Specifies if an application should overwrite the log file each time the application is restarted.

delimiter: Character which will be used as fields separator. Default value for DELIMITED output mode is ‘|’. Default value for “fixed widths” output mode is ‘ ’(space character).

trace: Specifies if trace mode is On (“YES” or “1”) or Off (“NO” or “0”). If trace is on, the log file will contain only trace messages.

minlevel: Indicates the minimum message level for log messages. Valid values are INFO, WARN, ERROR, or FATAL. The default is INFO.

lang: XXX, where XXX is a three letter language abbreviation that specifies the language of output text.

mode: Specifies one or more characteristics of the messages sent to the log file. The value can be set to the following values separated by commas:

- **RESTRICTED:** Log file will contain messages only from components specified in one ore more included **include** elements;
- **UNICODE:** output text for log file will be in UNICODE format;
- **PLAIN:** mode with no header information (component id, timestamp, level...) in log file;
- **DELIMITED:** output fields will have dynamic widths and they will be separated by delimiter.
- **NOUSERID:** information about the current user from Security Administration will be excluded from the output file.
- **FILEPERPROCESS:** Specifies that the logging framework create a single trace file for every running process. The name of the trace file adheres to the following naming convention:

<FileName>_<ExecutableName>_<ProcessID>_<DateTime>.<Extension>

where:

- <FileName> is the name of the trace file (specified in the “path” attribute)
- <ExecutableName> is the name of the executable
- <ProcessID> is the Windows process identifier
- <DateTime> is the date and time the file was created, represented as YYYY-MM-DD-HH24-MI-SS-SSS
- <Extension> is the extension specified for the file in the “path” attribute

For example, if the LSReportMonitor service was running along with 2 reports generated by LSReport.exe, the following files would be created in the ..\Lodestar\log directory:

- trace_lsreportmonitor_2944_2009-04-30-16-35-21-731.log
- trace_lsreport_324_2009-04-30-16-40-11-389.log
- trace_lsreport_4563_2009-04-30-16-40-15-435.log

processes: Indicates one or more processes (separated by semi-colons if needed) for which tracing/logging messages will be saved. The value of this attribute should be a process identifier in the form of an image name for a process as it appears in Windows Task Manager (java.exe, LSReportMonitor.exe, etc.) or an identifier that the application will pass to the logging framework (e.g. fileloader).

minlevel: Indicates the minimum message level for log messages. Valid values are INFO, WARN, ERROR, or FATAL. The default is INFO.

When tracing is enabled (the “trace” attribute is equal to “YES”), this indicates the level of tracing captured in the specified tracing file. Valid values include “INFO” and “VERBOSE”. The default is “INFO”. Trace files where this attribute is set to “VERBOSE” capture all tracing information and results in large tracing files that grow in size quickly. See **Tracing Levels** on page 11-29 for more information about tracing levels.

rotate: Indicates that new trace files be created when the current file reaches the size specified in this attribute. When the maximum size is reached by the current trace file, it will be renamed and a new trace file will be created.

The value of this attribute should be set according to the following XML Schema pattern rule: "[1-9]+[0-9]*[K|M|G]". Examples:

- 10K - ten kilobytes
- 99M - ninety nine megabytes
- 5G - five gigabytes

The names of trace files created adhere to the following naming convention:

<FileName>_<DateTime>.<Extension>

where:

- <FileName> is the name of the trace file (specified in the “path” attribute)
- <DateTime> is the date and time the file was renamed, represented as YYYY-MM-DD-HH24-MI-SS
- <Extension> is the extension specified for the file in the “path” attribute

For example, if this attribute was set to 100MB (100M), the following trace files would be created in the ..\Lodestar\log directory:

- trace.log (file with a size less than 100 Mb)
- trace_2009-04-30-16-35-21.log (file with the size ~100Mb)
- trace_2009-04-29-23-49-15.log (file with the size ~100Mb)

textwidth: Specifies the number of characters per line in the log file

Elements:

include: Element describes components to be included in the log file when in “RESTRICTED” mode.

Attributes:

compid: ID for the specified component to be included. See **Component Names and IDs** for a list of component id values.

minlevel: Indicates the minimum message level for log messages for the included component. Valid values are INFO, WARN, ERROR, or FATAL. The default is INFO.

exclude: Element describes components to be excluded from the log file when not in “RESTRICTED” mode.

Attributes:

compid: ID for the specified component to be excluded. See **Component Names and IDs** for a list of component id values.

systemlog: Element that defines how messages are written to an Event Log.

Attributes:

servername: Universal Naming Convention (UNC) name of the server with destination Event Log. By default this is the local computer.

minlevel: Indicates the minimum message level for system log messages. Valid values are INFO, WARN, ERROR, or FATAL. The default is INFO.

lang: XXX, where XXX is a three letter language abbreviation that specifies the language of output text.

trace: Specifies if trace mode is On (“YES” or “1”) or Off (“NO” or “0”). If trace is on, the Event Log will contain only trace messages.

mode: Specifies one or more characteristics of the messages sent to the Event Log. The value can be set to the following values separated by commas:

- **RESTRICTED:** Event Log will contain messages only from components specified in one ore more included **include** elements;
- **UNICODE:** output text for Event Log will be in UNICODE format;
- **PLAIN:** mode with no header information (component id, timestamp, level...) in Event Log;
- **DELIMITED:** output fields will have dynamic widths and they will be separated by delimiter.
- **NOUSERID:** information about the current user from Security Administration will be excluded from the output file.

processes: Indicates one or more processes (separated by semi-colons if needed) for which tracing/logging messages will be saved. The value of this attribute should be a process identifier in the form of an image name for a process as it appears in Windows Task Manager (java.exe, LSReportMonitor.exe, etc.) or an identifier that the application will pass to the logging framework (e.g. fileloader).

minlevel: Indicates the minimum message level for log messages. Valid values are INFO, WARN, ERROR, or FATAL. The default is INFO.

When tracing is enabled (the “trace” attribute is equal to “YES”), this indicates the level of tracing captured. Valid values include “INFO” and “VERBOSE”. The default is “INFO”. Setting this attribute to “VERBOSE” captures all tracing information. See **Tracing Levels** on page 11-29 for more information about tracing levels.

Elements:

include: Element describes components to be included in the Event Log when in “RESTRICTED” mode.

Attributes:

compid: ID for the specified component to be included. See **Component Names and IDs** for a list of component id values.

minlevel: Indicates the minimum message level for log messages for the included component. Valid values are INFO, WARN, ERROR, or FATAL. The default is INFO.

exclude: Element describes components to be excluded from the Event Log when not in “RESTRICTED” mode.

Attributes:

compid: ID for the specified component to be excluded. See **Component Names and IDs** for a list of component id values.

LSLOGGER.CFG.XML Example - All applications share one log file

```
<lslogger>
  <file path="lslog.log"/>
</lslogger>
```

LSLOGGER.CFG.XML Example - All applications trace in one file

```
trace_in_one_file
<lslogger>
  <file path="lstrace.log" trace="YES" />
</lslogger>
```

LSLOGGER.CFG.XML Example - All applications share one (Plain) file

```
<lslogger>
  <file path="lslog.log" mode="PLAIN"/>
</lslogger>
```

LSLOGGER.CFG.XML Example - One log file per component

```
<lslogger>
  <file path="lslog1.log" mode="RESTRICTED">
    <include compid="1"/>
  </file>
  <file path="lslog2.log" mode="RESTRICTED">
    <include compid="2"/>
  </file>
</lslogger>
```

LSLOGGER.CFG.XML Example - One log file per format

```
<lslogger>
  <file path="lslog_u.log" mode="UNICODE"/>
  <file path="lslog_a.log"/>
</lslogger>
```

LSLOGGER.CFG.XML Example - Use System Log on Local Machine

```
<lslogger>
  <systemlog/>
</lslogger>
```

LSLOGGER.CFG.XML Example - Mix of Log Files and Modes

```
<lslogger>
  <file path="lslog_i.log" mode="RESTRICTED" >
    <include compid="1" />
  </file>
  <file path="lslog_w.log" mode="PLAIN,UNICODE" />
  <file path="trace_LSDB4.log" trace="YES" mode="RESTRICTED" textwidth="300">
    <include compid="4" />
  </file>
</lslogger>
```

LSLOGGER.CFG.XML Example - Separate trace file per process

```
<lslogger>
  <file path="trace.log" trace="YES" mode="FILEPERPROCESS"/>
</lslogger>
```

LSLOGGER.CFG.XML Example - Specifying different tracing levels

```
<lslogger>
  <file path="trace_info.log" trace="YES"/>
  <file path="trace_verbose.log" trace="YES" minlevel="VERBOSE"/>
</lslogger>
```

LSLOGGER.CFG.XML Example - Specifying maximum file size

```
<lslogger>
  <file path="trace.log" trace="YES" rotate="100M"/>
</lslogger>
```

LSLOGGER.CFG.XML Example - Prevent runtime changes in configuration

```
<lslogger runtime_change="NO">
  <file path="log.log"/>
  <file path="trace.log" trace="YES"/>
</lslogger>
```

LSLOGGER.CFG.XML Example - Create dump file on error

```
<lslogger dump_on_se="YES">
  <file path="log.log"/>
  <file path="trace.log" trace="YES"/>
</lslogger>
```

Component Names and IDs

Component Name	Component ID
LSUTILS	1
LSRDB	2
LSRF	3
LSDB4	4
LSMETADB4	5
LSINTD	6
LSEProvider	7
BTEProvider	8
LSUPLOAD	9
LSSECURE	10
RDBProvider	11
LSDOC	12
XMLProvider	14
LSLIST	20
PLBASENT	30, 31
PLDBNT	32, 33

Component Name	Component ID
LSDB	34, 35
INTDWNT	36, 37
RXMDLNT	38, 39, 40
LSANAL	41, 42
LSACCT	43, 44, 45
PLVIEW	46, 47
REVFCST	48
LSWRKFLW	49
LSWorkQueue	50
LSCalendar	51
CollectionObjects	52
LSRulesSchedule	53
LSImport	54
Adapter	55,56

LSRELAY.CFG.XML

The LSRELAY.CFG.XML file identifies the machine running the SMTP service used to send email messages from the Oracle Utilities Energy Information Platform. This file is required if the email functions are used. The LSRELAY.CFG.XML file must be installed in the **C:\LODESTAR\CFG** directory.

A sample of this file is provided in the **C:\LODESTAR\CFG\Examples\CFG** directory.

LSRELAY.CFG.XML Example

```
<LSRELAY>
  <EMAIL>
    <EMAIL_SERVER>bxqasrv5</EMAIL_SERVER>
    <EMAIL_PORT>25</EMAIL_PORT>
    <EMAIL_AUTH>LOGIN</EMAIL_AUTH>
    <EMAIL_USER>Administrator</EMAIL_USER>
    <EMAIL_PASS>lodestar</EMAIL_PASS>
    <EMAIL_TIMEOUT>10000</EMAIL_TIMEOUT>
  </EMAIL>
</LSRELAY>
```

LSRELAY.CFG.XML Element Descriptions

Each of the data elements used by the LSRELAY.CFG.XML file are described below.

LSRELAY: Root element for the file that contains a EMAIL element.

EMAIL: Root element that defines the machine where the SMTP service used to send email is located.

Elements:

EMAIL_SERVER: The name (or IP address) of the machine where the SMTP service is located. See **Configuring an SMTP Email Server (Optional)** on page 4-15 in the *Oracle Utilities Energy Information Platform Installation Guide* for more information about setting up an SMTP email server.

EMAIL_PORT: the port number used by the email server. The default port is 25.

EMAIL_AUTH: The security mechanism used to connect to the email server. LOGIN is the only security mechanism supported.

EMAIL_USER: A valid Windows user ID with privileges on the email server. This element can be encrypted using the EncryptCFG.exe program. See **Encrypting Configuration Files** on page 2-54 for more information.

EMAIL_PASS: Password for the user ID specified in the EMAIL_USER element. This element can be encrypted using the EncryptCFG.exe program. See **Encrypting Configuration Files** on page 2-54 for more information.

EMAIL_TIMEOUT: The amount of time (in milliseconds) before which an unresponsive request to an email server times out and returns an error (Default: 2000, or 2 seconds).

LSREPORTMONITOR.CFG.XML

The LSREPORTMONITOR.CFG.XML file specifies how reports are processed through the web-enabled Energy Information Platform and related products by the LSReportMonitor service (LSREPORTMONITOR.EXE). This service monitors the Report Instance (LSRFREPORTINSTANCE) table in each data source specified in this file. The LSREPORTMONITOR.CFG.XML file must be installed in the C:\LODESTAR\CFG directory.

A sample of this file is provided in the C:\LODESTAR\CFG\Examples\CFG directory.

LSREPORTMONITOR.CFG.XML Example

```
<LSREPORTMONITOR>
  <DATASOURCE>
    <NAME>YourDatabaseName_1</NAME>
    <CONNECTSTRING>Data Source=TS92;User
ID=ls470bx;Password=password;LSProvider=ODP</CONNECTSTRING>
    <QUALIFIER>DatabaseQualifier</QUALIFIER>
  </DATASOURCE>
  <ALTERNATES>
    <CONNECTSTRING
NAME="YourDatabaseName_1">DSN=YourDSN;UID=YourUserName;PWD=YourPassword;</
CONNECTSTRING>
    <CONNECTSTRING NAME="YourDatabaseName_1"
REPORTTYPE="BIPublisher">jdbc:oracle:thin:@Host:Port:ServerName</
CONNECTSTRING>
  </ALTERNATES>
  <PROCESS_POOL_SIZE>5</PROCESS_POOL_SIZE>
  <PRIORITY_QUEUEING />
  <LOOP_TIMEOUT>5000</LOOP_TIMEOUT>
  <REPORT_TYPES>
    <REPORT_TYPE NAME="PLBX" PATH="C:\LODESTAR\BIN\lsreport.exe"/>
    <REPORT_TYPE NAME="LSRate" PATH="C:\LODESTAR\BIN\lsreport.exe"/>
    <REPORT_TYPE NAME="BIPublisher" PATH="C:\LODESTAR\BIN\LSBIReport.exe"/>
    <REPORT_TYPE NAME="Crystal" PATH="C:\LODESTAR\BIN\crgenreport.exe"
ALTERNATE_CS="true"/>
    <REPORT_TYPE NAME="LoadAnalysis"
PATH="C:\LODESTAR\LODESTAR110\BINS\csRptGenerator.exe"/>
    <REPORT_TYPE NAME="LSIMPORT" PATH="C:\LODESTAR\BIN\lsimport.exe"
ALTERNATE_CS="true"/>
  </REPORT_TYPES>
  <REPORT_OUTPUT_FORMATS>
    <OUTPUTFORMAT REPORTTYPE="BIPublisher" RPTOUTFMT="PortableDocFormat"
ENABLED="true"/>
    <OUTPUTFORMAT RPTOUTFMT="Excel80" ENABLED="false"/>
    <OUTPUTFORMAT RPTOUTFMT="XML" ENABLED="true"/>
  </REPORT_OUTPUT_FORMATS>
  <BI_PUBLISHER>
    <URL>http://localhost:8088/xmlpserver</URL>
    <CONNECTSTRING>UID=BIP userid;PWD=BIP password</CONNECTSTRING>
    <VERSION>10</VRESION>
    <TIMEOUT>500000</TIMEOUT>
  </BI_PUBLISHER>
</LSREPORTMONITOR>
```

LSREPORTMONITOR.CFG.XML Element Descriptions

Each of the data elements used by the LSREPORTMONITOR.CFG.XML file are described below.

LSREPORTMONITOR: Root element for the file.

DATASOURCE: Root element for a data source that stores report data and that is monitored by the LSReportMonitor service.

Elements:

NAME: The name of the data source.

CONNECTSTRING: The connect string for the Oracle Utilities Data Repository, in one of the following formats:

For Oracle databases:

```
<CONNECTSTRING>Data Source=<data_source>;User
ID=<user_id>;Password=<password>;LSProvider=ODP;</CONNECTSTRING>
```

For Microsoft SQL databases:

```
<CONNECTSTRING>Data Source=<address>;Initial
Catalog=<SQL_database>;User
Id=<user_id>;Password=<password>;MultipleActiveResultSets=True;LSP
rovider=MSSQL;</CONNECTSTRING>
```

where:

<data_source> is the Oracle TNS Name for the data source, from the TNS_NAMES.ora file (typically located in the \\<machine>\oracle\network\admin directory)

<user_id> is the user ID for the database connection

<password> is the password for the supplied user ID.

<address> is the IP address or Hostname of the MS SQL Server database server

<SQL_database> is the name of the MS SQL Server database

QUALIFIER: The qualifier or schema used for database object access (tables, views, etc.).

ALTERNATES: Root element for alternate ODBC data sources. Required when using Oracle BI Publisher or Crystal Reports.

CONNECTSTRING: The connect string for the Oracle Utilities Data Repository. Can be one of the following formats:

For Crystal Reports:

```
<CONNECTSTRING>DSN=<data_source>;UID=<user_id>;PWD=<password>;</
CONNECTSTRING>
```

where:

<data_source> is the DSN for the Oracle Utilities Data Repository

<user_id> is the user ID for the database connection

<password> is the password for the supplied user ID.

For BI Publisher Reports:

For Oracle:

```
<CONNECTSTRING NAME="DBName"
REPORTTYPE="BIPublisher">jdbc:oracle:thin:@<Host>:<Port>:
<O_Database></CONNECTSTRING>
```

For MS SQL Server:

```
<CONNECTSTRING NAME="DBName" REPORTTYPE="BiPublisher">{JDBC_URL} //
<Servername>/<SQL_Database></CONNECTSTRING>
```

where:

<Host> is the name of the database server machine

<Port> is the port the database connection will use. Default is 1521.

<O_Database> is the Oracle TNS Name for the data source, from the TNS_NAMES.ora file (typically located in the \\<machine>\oracle\network\admin directory)

{JDBC_URL} is the URL of the JDBC driver. Consult your driver's documentation for proper composition.

<ServerName> is the password for the supplied user ID.

<SQL_Database> is the name of the MS SQL Server database

Attributes:

NAME: The name of the data source (defined in the DATASOURCE element) for which this is an alternate. This must match the value of the NAME element for the data source for which this is an alternate.

REPORTTYPE: The report type for which this alternate connection is used. The default value for this attribute is "Crystal." This attribute is required only with Oracle BI Publisher reports, and should have a value of "BiPublisher."

PROCESS_POOL_SIZE: The maximum number of reports that can be run simultaneously on the server. This should be set to 2 to 3 times the number of processors on the server. For example, a quad processor server should have a PROCESS_POOL_SIZE ranging from 8 to 12. The default value is 5.

PRIORITY_QUEUEING: If this element is present, the LSReportMonitor service selects reports to process based on assigned priorities (High, Medium, or Low). Priorities can be assigned to report templates using the Report Administration user interface, and can be set for specific report instances at runtime.

LOOP_TIMEOUT: Time interval between working cycles, in milliseconds. The default value is 30,000 (30 seconds).

REPORT_TYPES: The types of reports.

Elements:

REPORT_TYPE: A specific report type.

Attributes:

NAME: The name of the report type. Valid values and their corresponding programs are:

- PLBX/LSREPORT.EXE (Oracle Utilities Billing Component reports and analyses)
- LSRate/LSREPORT.EXE (Rules Language processing)
- LSRUNRs/LSRUNRATE.EXE (Rules Language validations and mappings in Oracle Utilities Adapter)
- Crystal/CRGENREPORT.EXE (Crystal Reports)
- Active/LSREPORT.EXE (Active Reports, used by Oracle Utilities Receivables Component)
- LSIMPORT/LSIMPORT.EXE (Data import)

- BIPublisher/LSBIReport.exe (Oracle BI Publisher reports)
- LoadAnalysis/CsRptGenerator.exe (Oracle Utilities Load Analysis Programs)

PATH: The path to the program (LSREPORT.EXE, LSIMPORT.EXE, etc.) used by the report type. This is usually installed in the C:\LODESTAR\BIN directory.

ALTERNATE_CS: Indicates if using an alternate data source for Oracle BI Publisher or Crystal Reports. Used only when the NAME attribute is equal to “BIPublisher” or “Crystal.” Valid values are “true” and “false.” If this is set to true, an alternate data source is required for each data source defined in this file.

REPORT_OUTPUT_FORMATS: Element that defines output formats for Oracle BI Publisher or Crystal Reports. Inclusion of this element overrides the default settings for available output formats. See **Report Output Format Details** on page 2-39 for more information.

Elements:

OUTPUTFORMAT: A specific output format.

Attributes:

REPORTTYPE: The report type for which this alternate connection is used. This attribute is used only with Oracle BI Publisher reports, and should have a value of “BiPublisher.”

RPTOUTFMT: The name of the output format. Valid formats and their corresponding values are:

- Adobe Acrobat (“PortableDocFormat”)
- Microsoft Excel 97-2000 (“Excel80”)
- Microsoft Excel 97-2000-Data Only (“Excel80Tabular”)
- Microsoft Word (“WordForWindows”)
- Rich Text Format (“ExactRichText”)
- Separated Values - CSV (“CommaSeparatedValues”)
- XML (“XML”)
- Microsoft PowerPoint (“PowerPoint”) - Oracle BI Publisher only
- Microsoft Excel (“Excel2000”) - Oracle BI Publisher only
- Web Archive (“MHTML”) - Oracle BI Publisher only

ENABLED: Indicates if the output format is enabled. Can be “true” or “false”.

ISDEFAULT: Indicates if the output format is the default format for the first appearance of reports generated with an unspecified output format. Can be “true” or “false”.

BI_PUBLISHER: Element that defines connection to Oracle BI Publisher server. Required when using Oracle BI Publisher.

Elements:

URL: The URL for the Oracle BI Publisher server, in the following format:

```
<URL>http://<server>:<port>/xmlpserver</URL>
```

where:

<server> is the machine name or IP address of the server on which the Oracle BI Publisher server is installed

<port> is the port used to connect to the Oracle BI Publisher server

CONNECTSTRING: The connection string for the Oracle BI Publisher server in the following format:

```
<CONNECTSTRING>UID=<user_id>;PWD=<password>;</CONNECTSTRING>
```

where:

<user_id> is the user ID for the connection to the Oracle BI Publisher server. This user ID must be set up on the Oracle BI Publisher server.

<password> is the password for the supplied user ID.

VERSION: The version of Oracle BI Publisher installed and being used, in the following format:

```
<VERSION>10</VERSION>
```

This release supports Oracle Business Intelligence Publisher version 10.1.3.4 and version 11.1.1.

- To specify use of Oracle Business Intelligence Publisher version 10.1.3.4, enter a value of “10” in the <VERSION> element.
- To specify use of Oracle Business Intelligence Publisher version 11.1.1, enter a value of “11” in the <VERSION> element.

TIMEOUT: The amount of time, in milliseconds, to allow the report manager to establish a connection to the Business Intelligence Publisher server before returning a timeout error. There is no default value. To specify that there should never be a timeout, enter a value “INFINITE” for this element.

Report Output Format Details

The following table lists details associated with each of the output formats available for Oracle BI Publisher and Crystal Reports.

Format	Format Code	Ext.	MIME type	Default for 1st	Default
Adobe Format	PortableDocFormat	.pdf	application/pdf	Yes	Enabled
Microsoft Excel	Excel80	.xls	application/vnd.ms-excel	No	Enabled
Microsoft Excel (Data only)	Excel80Tabular	.xls	application/vnd.ms-excel	No	Disabled
Microsoft Word	WordForWindows	.doc	application/msword	No	Enabled
Rich Text Format	ExactRichText	.rtf	application/rtf	No	Enabled
Separated Values	CommaSeparatedValues	.csv	text/csv	No	Disabled
XML	XML	.xml	text/xml	No	Disabled
Web Archive	MHTML	.mhtml	text/mhtml	No	Disabled
Microsoft PowerPoint	PowerPoint	.ppt	text/ppt	No	Disabled
Microsoft Excel 2000	Excel2000	.xls	application/vnd.ms-excel	No	Disabled

For each format, this table lists the following:

Format: the name of the output format.

Format Code: the code used to designate the output format in the LSREPORTMONITOR.CFG.XML file. Used in the RPTOUTFMT attribute of the OUTPUTFORMAT element.

Ext.: The file extension associated with the output format.

MIME Type: the document MIME type associated with the output format.

Default for 1st: indicates if the output format is the default for the first appearance of reports generated with an unspecified output format. Used in the ISDEFAULT attribute of the OUTPUTFORMAT element.

Default: indicates if the output format is enabled by default.

LSSCHDLR.CFG.XML

The LSSCHDLR.CFG.XML file defines the data source(s) monitored by the Oracle Utilities Schedule Service (LSSCHDLR.exe). This service monitors the Scheduled Message (SCHEDULEDMESSAGE) table in each data source specified in this file. This file is required if the Oracle Utilities Schedule Service is used. The LSSCHDLR.CFG.XML file must be installed in the **C:\LODESTAR\CFG directory**.

A sample of this file is provided in the **C:\LODESTAR\CFG\Examples\CFG** directory.

LSSCHDLR.CFG.XML Example

```
<LSSHDLR>
  <DATASOURCE>
    <NAME>DatabaseName</NAME>
    <CONNECTSTRING>Data Source=TS92;User
ID=ls470bx;Password=password;LSProvider=ODP</CONNECTSTRING>
    <QUALIFIER>DatabaseQualifier</QUALIFIER>
  </DATASOURCE>
  <LOOP_TIMEOUT>5000</LOOP_TIMEOUT>
</LSSHDLR>
```

LSSCHDLR.CFG.XML Element Descriptions

Each of the data elements used by the LSSCHDLR.CFG.XML file are described below.

LSSHDLR: Root element for the file that contains one or more DATASOURCE elements.

DATASOURCE: Root element for the data source monitored by the Oracle Utilities Schedule Service (LSSCHDLR.exe).

Elements:

NAME: The name of the data source.

CONNECTSTRING: The connect string for the Oracle Utilities Data Repository, in one of the following formats:

For Oracle databases:

```
<CONNECTSTRING>Data Source=<data_source>;User
ID=<user_id>;Password=<password>;LSProvider=ODP;</CONNECTSTRING>
```

For Microsoft SQL databases:

```
<CONNECTSTRING>Data Source=<address>;Initial
Catalog=<SQL_database>;User
Id=<user_id>;Password=<password>;MultipleActiveResultSets=True;LSP
rovider=MSSQL;</CONNECTSTRING>
```

where:

- <data_source> is the Oracle TNS Name for the data source, from the TNS_NAMES.ora file (typically located in the \\<machine>\oracle\network\admin directory)
- <user_id> is the user ID for the database connection
- <password> is the password for the supplied user ID.
- <address> is the IP address or Hostname of the MS SQL Server database server
- <SQL_database> is the name of the MS SQL Server database

QUALIFIER: The qualifier or schema used for database object access (tables, views, etc.).

LOOP_TIMEOUT: Time interval between working cycles, in milliseconds.

LSSECURE.CFG.XML

The LSSECURE.CFG.XML file specifies the data source used by the Security functionality (required), an optional session expire time (in minutes), optional parameters used when running Security in LDAP mode (see **LDAP Mode** on page 12-3), optional parameters used to define password requirements, and optional parameters used to specify user account settings. The LSSECURE.CFG.XML file must be installed in the **C:\LODESTAR\CFG** directory.

A sample of this file is provided in the **C:\LODESTAR\CFG\Examples\CFG** directory.

LSSECURE.CFG.XML Example - Default Mode

```
<LSSECURE>
  <DATASOURCE>
    <NAME>DatabaseName</NAME>
    <CONNECTSTRING>Data Source=TS92;User
ID=ls470bx;Password=password;LSProvider=ODP</CONNECTSTRING>
    <QUALIFIER>DatabaseQualifier</QUALIFIER>
  </DATASOURCE>
  <EXPIRETIME>60</EXPIRETIME>
  <PASSWORDS>
    <MINLENGTH>8</MINLENGTH>
    <NUMLETTERS>1</NUMLETTERS>
    <NUMDIGITS>1</NUMDIGITS>
    <DIFFCASES>NO</DIFFCASES>
    <NUMPREVIOUS>6</NUMPREVIOUS>
    <EXPDAYS>180</EXPDAYS>
    <EXPDAYSHINT>10</EXPDAYSHINT>
    <CHANGEINIT>YES</CHANGEINIT>
  </PASSWORDS>
  <ACCOUNTS>
    <NUMFAILED>5</NUMFAILED>
    <INACTIVEDAYS>180</INACTIVEDAYS>
    <LOGINTRACKING>YES</LOGINTRACKING>
  </ACCOUNTS>
  <USERSINGLESESSION>YES</USERSINGLESESSION>
</LSSECURE>
```

LSSECURE.CFG.XML Example - LDAP Mode

```
<LSSECURE>
  <DATASOURCE>
    <NAME>DatabaseName</NAME>
    <CONNECTSTRING>Data Source=TS92;User
ID=ls470bx;Password=password;LSProvider=ODP</CONNECTSTRING>
    <QUALIFIER>DatabaseQualifier</QUALIFIER>
  </DATASOURCE>
  <EXPIRETIME>60</EXPIRETIME>
  <SSO FAILOVER="IGNORESSO">LOGON_USER</SSO>
  <LDAP>
    <SERVER>
      <HOST>dc1 ldap.lan</HOST>
      <VERSION>3</VERSION>
      <PORT>389</PORT>
      <TIMEOUT>30000</TIMEOUT>
      <UDN>CN=Administrator,CN=Users,DC=ldap,DC=lan</UDN>
      <PWD>...</PWD>
      <USERSEARCH>
        <BASEDN>CN=Users,DC=ldap,DC=lan</BASEDN>
        <SCOPE>ONE</SCOPE>
        <UIDATTRNAME>UID</UIDATTRNAME>
        <FILTER>(objectClass=user)</FILTER>
      </USERSEARCH>
    </SERVER>
  </LDAP>
</LSSECURE>
```

LSSECURE.CFG.XML Element Descriptions

Each of the data elements used by the LSSECURE.CFG.XML file are described below.

DATASOURCE: Root element for the data source that contains the security data (Data Sources, Groups, and Users).

Elements:

NAME: The name of the data source.

CONNECTSTRING: The connect string for the Oracle Utilities Data Repository, in one of the following formats:

For Oracle databases:

```
<CONNECTSTRING>Data Source=<data_source>;User
ID=<user_id>;Password=<password>;LSProvider=ODP;</CONNECTSTRING>
```

For Microsoft SQL databases:

```
<CONNECTSTRING>Data Source=<address>;Initial
Catalog=<SQL_database>;User
Id=<user_id>;Password=<password>;MultipleActiveResultSets=True;LSP
rovider=MSSQL;</CONNECTSTRING>
```

where:

- <data_source> is the Oracle TNS Name for the data source, from the TNS_NAMES.ora file (typically located in the \\<machine>\oracle\network\admin directory)
- <user_id> is the user ID for the database connection
- <password> is the password for the supplied user ID.
- <address> is the IP address or Hostname of the MS SQL Server database server
- <SQL_database> is the name of the MS SQL Server database

QUALIFIER: The qualifier or schema used for database object access (tables, views, etc.).

EXPIRETIME: Optional session expire time (in minutes). If not included, the expire time defaults to 60 minutes.

SSO: Defines a parameter used by Microsoft Internet Information Service (IIS) to pass the user's User ID to the security component for purposes of user authentication.

- To use Windows Integrated Authentication, specify "**LOGON_USER**" in the <SSO> element.

Note: IIS must be configured to support Windows Integrated Authentication. See **IIS Configuration** on page 12-30 for more information.

- To use an HTTP Header request parameter (used with SSO systems, such as SiteMinder), specify "**HTTP_<HEADER>**" in the <SSO> element, where <HEADER> is the name of the header request parameter used by the SSO system. For example, if using SiteMinder, you would specify "**HTTP_SMUSER**" in the <SSO> element.

Note: The <HEADER> parameter must NOT contain dashes (-) or underscore characters.

Attributes:

FAILOVER: Specifies the behavior in the event of failure when connecting to the user authentication directory server. To default to the current non-single sign-on user authentication method (LDAP mode or default mode, set this attribute equal to "IGNORESSO" (FAILOVER="IGNORESSO"). To fail authentication if SSO is missing or empty, omit this attribute or set this attribute equal to NULL (FAILOVER="").

SSOLOGOUT: Specifies a web page used as a landing page that single sign-on users are directed to when they log out of the application.

LDAP: Root element for an LDAP server that contains LDAP security data. This element is required when running security in LDAP mode, and must be omitted when running security in default mode.

Elements:

SERVER: Defines a single LDAP server.

Attributes:

GROUP: The name of the group of servers to which the server belongs. Groups of related servers are considered “mirrors” of each other, for purposes of redundancy. If this attribute is not specified, the server is considered to belong to an “empty” group.

Elements:

HOST: A fully qualified domain name (FQDN) or IP address of the LDAP server.

VERSION: The LDAP version. Oracle Utilities supports version 3.

PORT: The connection TCP port number used by the LDAP server. The Default value is 389.

TIMEOUT: The connection timeout in milliseconds. The Default value is 30000.

UDN: The DN (“distinguished name”) of the directory entry for the user, used for binding (login) to LDAP server. Absence of UDN indicates that the LDAP server supports anonymous access.

PWD: The password for the UDN above. Not required if UDN is absent. This password must be encrypted using the LSSecureInit.exe command line program. See **Setting Up Security in LDAP Mode** on page 12-20 for more information.

USERSEARCH: Element that defines how the security component will search for users on the LDAP server. The combination of the BASEDN, SCOPE, and FILTER values allow decreasing the number of directory entries accessed when searching users.

Elements:

BASEDN: The base DN (distinguished name). This is the DN of the directory entry at which searches begin when searching the LDAP database for users. Searches begin at this DN and progress down the directory hierarchy (as defined by the SCOPE element).

SCOPE: The depth from the BASEDN to which searches should occur. Valid values include:

BASE: Indicates that searches occur only on the entries at the BASEDN.

ONE: Indicates that searches occur only on entries one level under the BASEDN, but not including the BASEDN.

SUBTREE (default): indicates that searches occur on entries at all levels under and including the BASEDN.

UIDATTRNAME: The name of the “UID” attribute in the LDAP security schema. For example, when using Microsoft Active Director server, you can specify **sAMAccountName**. The value of this attribute is shared with the security database, and will be used by end users as the User ID on the Login screen. When searching the directory tree, this attribute is interpreted as

`<UIDATTRNAME>= "User ID from login screen"`

The default is UID.

FILTER: An optional filter that allows adding additional criteria to user searches. The value of this element is concatenated with the UIDATTRNAME value as follows:

```
(& (UIDATTRNAME)= "User ID from login screen") <FILTER> )
```

PASSWORDS: Root element for optional password related settings. If this element is present, a user's password cannot contain the user's User ID.

Elements:

MINLENGTH: Minimum length of password (default 1).

NUMLETTERS: Minimum number of letters in password.

NUMDIGITS: Minimum number of digits in password.

DIFFCASES: Indicates if presence of different cases is required. Can be "YES" or "NO."

NUMPREVIOUS: The number of previous passwords that a new password cannot equal. Example: A user has had 5 previous passwords, including current one. If this value is set to 5, his new password could not be the same as any of his previous passwords.

EXPDAYS: The number of days after which passwords expire. User IDs with expired passwords are considered "Disabled*."

EXPDAYSHINT: The number of days before the expiration date for the user's password that the user will be notified that his password is set to expire.

CHANGEINIT: Indicates that passwords assigned by administrators should be changed during the user's next session. Can be "YES" or "NO."

ACCOUNTS: Root element for optional user ID-related settings.

Elements:

NUMFAILED: The number of failed logon attempts after which the user ID is disabled*. A value of 0 disables this feature. (Default: 0)

INACTIVEDAYS: The number of days of user inactivity after which the user ID is disabled*. A value of 0 disables this feature. (Default: 0)

LOGINTRACKING: Specifies whether or not login attempts are tracked. Valid values are YES (track login attempts) or NO (do not track login attempts). Omitting this element disables this feature.

* Users with access to the **Security Admin** menu item (in the **Tools and Utilities** menu) are considered Administrators, and therefore can not be disabled automatically. This is to prevent administrative users from being disabled, leaving no users able to perform security administration tasks.

USERSINGLESESSION: Specifies whether or not users are restricted to single concurrent sessions. The only valid value is YES (allow only 1 session at a time per user). Omitting this element allows multiple concurrent sessions for the same user (default).

Important

This file **MUST** be edited to specify the security data source prior to running Oracle Utilities applications.

LTMH.CFG.XML

The LTMH.CFG.XML file defines the database used by the Oracle Utilities Adapter and the Oracle Utilities Transaction Management. This file is required if the Oracle Utilities Adapter or Oracle Utilities Transaction Management is used. The LTMH.CFG.XML file must be installed in the **C:\LODESTAR\CFG directory**.

A sample of this file is provided in the **C:\LODESTAR\CFG\Examples\CFG** directory. This sample file also contains the XML schema for this configuration file which can be used to validate the structure of the file.

LTMH.CFG.XML Example

```
<LTMH JMSENABLED="false">
  <!-- Uncomment and customize lines appropriate to your installation -->
  <!-- Override the default JNDI settings for finding JMS information -->
  <!--
  <SECURITY>
    <KEYSTORE><PASSWORD>lodestar</PASSWORD></KEYSTORE>
    <TRUST><PASSWORD>lodestar</PASSWORD></TRUST>
  </SECURITY>
  <JNDI>
    <CONTEXTFACTORY>com.sun.jndi.fscontext.RefFSContextFactory</CONTEXTFACTORY>
    <PROVIDERURL>file:///C:/LODESTAR/LTMH/Runtime/jndi</PROVIDERURL>
  </JNDI>
  -->
  <!-- Connection for an Oracle database -->
  <JDBC URL="jdbc:oracle:thin:@localhost:1521:orcl">
    <DRIVER TYPE="ORACLE">oracle.jdbc.OracleDriver</DRIVER>
    <QUALIFIER>tr450</QUALIFIER>
    <USERNAME>tr450</USERNAME>
    <PASSWORD>password</PASSWORD>
  </JDBC>
  <!-- Optional database connection information
    - Service property settings will override these
    - Uncomment the appropriate section(s) -->
  <DEFAULT>
    <ORACLECONNECTION>
      <DSN>local</DSN>
      <QUALIFIER>tr450</QUALIFIER>
      <USERNAME>tr450</USERNAME>
      <PASSWORD>password</PASSWORD>
    </ORACLECONNECTION>
  </DEFAULT>
</LTMH>
```

LTMH.CFG Element Descriptions

Each of the elements used by the LTMH.CFG.XML file are described below.

LTMH: Root element that contains database connection information.

Attributes:

JMSENABLED: Indicates if the Oracle Utilities Adapter (or Oracle Utilities Transaction Management) will use Java Message Service (JMS) servers ("TRUE" or "FALSE").

RESTARTINTERVAL: Defines the amount of time (in minutes) after which services will be stopped. Once the specified time has been reached, the service will stop. If the service has been configured to restart (KEEPALIVE = "Y" in the Service Activation table), the service will automatically restart.

AUTO_REFRESH_SERVICES: Indicates if Adapter Runtime Services should automatically refresh in the Adapter Monitor. Valid values are "true" and "false" (default).

SERVICEPROFILINGENABLED=: Indicates if periodic logging of the number of running Adapter Runtime Services and their individual memory profiles should be included

in the Adapter/Transaction Management log files. The interval at which this logging is captured is defined by the **SERVICEPROFILINGINTERVAL** attribute. Valid values are "true" and "false" (default).

SERVICEPROFILINGINTERVAL=: Specifies the amount of time (in seconds) between memory profile logging. Valid only if the **SERVICEPROFILINGENABLED** attribute is set to "true". Default value is 60 (1 minute).

NETWORKCOMMUNICATIONENABLED=: If set to "true", network communication between the Adapter/Transaction Management server and Runtime Services is enabled. When network communication is enabled, network communication operational logs are created. By default, these are included in the main logs as well as in separate log files (with "NETWORK" appended to the log file name), but this can be overridden using the **NETWORKLOGTOFILEONLY** attribute. Valid values are "true" and "false" (default).

NETWORKLOGTOFILEONLY=: If set to "true", inclusion of network communication operational logs in the main Adapter/Transaction Management logs is disabled (by default, these are included in the main logs as well as in a separate log files). Valid values are "true" and "false" (default).

Elements:

SECURITY: Elements that define passwords for accessing the keystore and truststore files used when securing communications between the Adapter Server and Adapter Monitor. See **Securing and Encrypting the Adapter Server and Monitor on page 6-79** for more information.

Elements:

KEYSTORE: Element that specifies the password for the keystore file.

Elements:

PASSWORD: The password used to access the keystore file.

TRUST: Element that specifies the password for the truststore file.

Elements:

PASSWORD: The password used to access the truststore file.

JNDI: Elements that define a JDBC connection to the Oracle Utilities Data Repository.

Elements:

CONTEXTFACTORY: Defines the JNDI Context Factory class name used to identify the kind of JNDI server in order to retrieve JMS context information. If this value is not defined in this file, the Java System property "java.naming.factory.initial" is used. If this system property is not defined, the default setting is "com.sun.jndi.fscontext.RefFSContextFactory". This default is backward compatible with previous versions.

PROVIDERURL: Defines the JNDI Provider URL used to identify where the JNDI server is located in order to retrieve JMS context information. If this value is not defined in this file, the Java System property "java.naming.provider.url" is used. If the system property is not defined, the default setting is the "jndi" folder relative to the runtime folder of LTMH (usually lodestar/ltmh/runtime/jndi). This default is backward compatible with previous versions.

JDBC: Elements that define a JDBC connection to the Oracle Utilities Data Repository.

Attributes:

URL: defines the database used by the Adapter (or Oracle Utilities Transaction Management), in the following format:

For Oracle:

`jdbc:oracle:thin:@<ServerName>:<Port>:<YourDatabaseName>`

where:

- **<ServerName>** is the machine name of the database server.
- **<Port>** is the port the database connection will use. Default is 1521.
- **<YourDatabaseName>** is the TNS service name used by the database's ODBC connection.

For MS SQL Server:

```
jdbc:jtds:sqlserver://<ServerName>/<Database>
```

where:

- **<ServerName>** is the machine name of the database server.
- **<Database>** is the database name.

Elements:

DRIVER: the provider used to connect to the database. This is dependent on the type of database. The provider used with Oracle is "oracle.jdbc.OracleDriver". The provider used with MS SQL is "net.sourceforge.jtds.jdbc.Driver."

Attributes:

TYPE: The type of database. Can be either ORACLE or SQLSERVER.

QUALIFIER: Database qualifier. Must be UPPER case.

USERNAME: User ID to connect to the database. If using an alternative qualifier, this must be the qualifier.

PASSWORD: Password used to connect to the database.

RDL Database Connection Information

The following elements can be used to define database connections for all Adapter services. These parameters correspond to Runtime Service properties, and if supplied, Runtime Service properties override these parameters.

DEFAULT: Defines default database connections for all Runtime Services.

Elements:

SESSION: Defines user information for session connections. Uses a web user ID to establish a database connection. Used in the place of either the ORACLECONNECTION or SQLSERVERCONNECTION elements.

Elements:

USERNAME: User ID to connect to the database.

PASSWORD: Password used to connect to the database.

ALTERNATEDATASOURCE: optional name for a secondary data source for the web user.

ORACLECONNECTION: Defines default direct database connection information when connecting to an Oracle database. Used in the place of the SESSION element.

Elements:

DATASOURCE: Global setting for **DEFAULT_DSN** on page 11-34 of the *Oracle Utilities Energy Information Platform User's Guide*.

QUALIFIER: Global setting for **DEFAULT_DDBQUALIFIER** on page 11-34 of the *Oracle Utilities Energy Information Platform User's Guide*. Must be UPPER case.

USERNAME: Global setting for **DEFAULT_USER** on page 11-34 of the *Oracle Utilities Energy Information Platform User's Guide*.

PASSWORD: Global setting for **DEFAULT_PASSWORD** on page 11-35 of the *Oracle Utilities Energy Information Platform User's Guide*.

SQLSERVERCONNECTION: Defines default direct database connection information when connecting to a Microsoft SQL Server database. Used in the place of the SESSION element

Elements:

DATASOURCE: Global setting for **DEFAULT_DSN** on page 11-34 of the *Oracle Utilities Energy Information Platform User's Guide*.

INITIAL_CATALOG: Global setting for **DEFAULT_INITIALCATALOG** on page 11-34 of the *Oracle Utilities Energy Information Platform User's Guide*.

QUALIFIER: Global setting for **DEFAULT_DDBQUALIFIER** on page 11-34 of the *Oracle Utilities Energy Information Platform User's Guide*. Must be UPPER case.

USERNAME: Global setting for **DEFAULT_USER** on page 11-34 of the *Oracle Utilities Energy Information Platform User's Guide*.

PASSWORD: Global setting for **DEFAULT_PASSWORD** on page 11-35 of the *Oracle Utilities Energy Information Platform User's Guide*.

ADAPTER: Defines Business Rule source (RDL1) and Business Rule destination (RDL2) database connections for all Runtime Services.

Elements:

SESSION: Defines user information for session connections. Uses a web user ID to establish a database connection. Used in the place of the CONNECTION element.

Elements:

USERNAME: User ID to connect to the database.

PASSWORD: Password used to connect to the database.

ALTERNATEDATASOURCE: optional name for a secondary data source for the web user.

CONNECTION: Defines Business Rule source (RDL1) and Business Rule destination (RDL2) database connection information. Used in the place of the SESSION element.

Elements:

RDL1: Defines Business Rule Source (RDL1) database connection information.

Elements:

ORACLECONNECTION: Defines direct Business Rule source (RDL1) database connection information when connecting to Oracle databases.

Elements:

DATASOURCE: Global setting for **RDL_DSN_1** on page 11-37 of the *Oracle Utilities Energy Information Platform User's Guide*.

QUALIFIER: Global setting for **RDL_DDBQUALIFIER_1** on page 11-37 of the *Oracle Utilities Energy Information Platform User's Guide*. Must be UPPER case.

USERNAME: Global setting for **RDL_USER_1** on page 11-37 of the *Oracle Utilities Energy Information Platform User's Guide*.

PASSWORD: Global setting for **RDL_PASSWORD_1** on page 11-38 of the *Oracle Utilities Energy Information Platform User's Guide*.

SQLSERVERCONNECTION: Defines direct Business Rule source (RDL1) database connection information when connecting to Microsoft SQL Server databases.

Elements:

DATASOURCE: Global setting for **RDL_DSN_1** on page 11-37 of the *Oracle Utilities Energy Information Platform User's Guide*.

INITIAL_CATALOG: Global setting for **RDL_INITIALCATALOG_1** on page 11-37 of the *Oracle Utilities Energy Information Platform User's Guide*.

QUALIFIER: Global setting for **RDL_DDBQUALIFIER_1** on page 11-37 of the *Oracle Utilities Energy Information Platform User's Guide*. Must be UPPER case.

USERNAME: Global setting for **RDL_USER_1** on page 11-37 of the *Oracle Utilities Energy Information Platform User's Guide*.

PASSWORD: Global setting for **RDL_PASSWORD_1** on page 11-38 of the *Oracle Utilities Energy Information Platform User's Guide*.

RDL2: Defines direct Business Rule Destination (RDL2) database connection information.

Elements:

ORACLECONNECTION: Defines direct Business Rule destination (RDL2) database connection information when connecting to Oracle databases.

Elements:

DATASOURCE: Global setting for **RDL_DSN_2** on page 11-38 of the *Oracle Utilities Energy Information Platform User's Guide*.

QUALIFIER: Global setting for **RDL_DDBQUALIFIER_2** on page 11-39 of the *Oracle Utilities Energy Information Platform User's Guide*. Must be UPPER case.

USERNAME: Global setting for **RDL_USER_2** on page 11-39 of the *Oracle Utilities Energy Information Platform User's Guide*.

PASSWORD: Global setting for **RDL_PASSWORD_2** on page 11-39 of the *Oracle Utilities Energy Information Platform User's Guide*.

SQLSERVERCONNECTION: Defines direct Business Rule destination (RDL2) database connection information when connecting to Microsoft SQL Server databases.

Elements:

DATASOURCE: Global setting for **RDL_DSN_2** on page 11-38 of the *Oracle Utilities Energy Information Platform User's Guide*.

INITIAL_CATALOG: Global setting for **RDL_INITIALCATALOG_2** on page 11-38 of the *Oracle Utilities Energy Information Platform User's Guide*.

QUALIFIER: Global setting for **RDL_DDBQUALIFIER_2** on page 11-39 of the *Oracle Utilities Energy Information Platform User's Guide*. Must be UPPER case.

USERNAME: Global setting for **RDL_USER_2** on page 11-39 of the *Oracle Utilities Energy Information Platform User's Guide*.

PASSWORD: Global setting for **RDL_PASSWORD_2** on page 11-39 of the *Oracle Utilities Energy Information Platform User's Guide*.

WORKQUEUE: Defines work queue database connections for all Runtime Services.

Elements:

SESSION: Defines user information for session connections. Uses a web user ID to establish a database connection. Used in the place of the CONNECTION element.

Elements:

USERNAME: User ID to connect to the database.

PASSWORD: Password used to connect to the database.

ALTERNATEDATASOURCE: optional name for a secondary data source for the web user.

ORACLECONNECTION: Defines direct work queue database connection information when connection to Oracle databases. Used in the place of the SESSION element.

Elements:

DATASOURCE: Global setting for **WQ_DSN** on page 11-49 of the *Oracle Utilities Energy Information Platform User's Guide*.

QUALIFIER: Global setting for **WQ_DDBQUALIFIER** on page 11-49 of the *Oracle Utilities Energy Information Platform User's Guide*. Must be UPPER case.

USERNAME: Global setting for **WQ_USER** on page 11-49 of the *Oracle Utilities Energy Information Platform User's Guide*.

PASSWORD: Global setting for **WQ_PASSWORD** on page 11-50 of the *Oracle Utilities Energy Information Platform User's Guide*.

SQLSERVERCONNECTION: Defines direct work queue database connection information when connecting to Microsoft SQL Server databases. Used in the place of the SESSION element.

Elements:

DATASOURCE: Global setting for **WQ_DSN** on page 11-49 of the *Oracle Utilities Energy Information Platform User's Guide*.

INITIAL_CATALOG: Global setting for **WQ_INITIALCATALOG** on page 11-49 of the *Oracle Utilities Energy Information Platform User's Guide*.

QUALIFIER: Global setting for **WQ_DDBQUALIFIER** on page 11-49 of the *Oracle Utilities Energy Information Platform User's Guide*. Must be UPPER case.

USERNAME: Global setting for **WQ_USER** on page 11-49 of the *Oracle Utilities Energy Information Platform User's Guide*.

PASSWORD: Global setting for **WQ_PASSWORD** on page 11-50 of the *Oracle Utilities Energy Information Platform User's Guide*.

Other Configuration Files

Your Oracle Utilities application software uses other configuration files, including the **Tools Files (*.TLS)** file, described below.

Tools Files (*.TLS)

The *.TLS file allows you to make other Oracle Utilities products, or even third-party utilities and applications, accessible from the menu bar of a Oracle Utilities application program

For example, you might make Oracle Utilities Billing Component, Microsoft Excel, and a calendar accessory available from within Data Manager. After you had modified the DATAMGR.TLS file, a user could open any of these programs while working in Data Manager by selecting **Tools->[program-name]**.

A sample *.tls file is provided in the **C:\LODESTAR\CFG\Examples\CFG** directory.

How to Modify the .TLS file:

1. Go to the LODESTAR\CFG directory.
2. Using Notepad or another editor, open the .TLS file for the Oracle Utilities product whose menu bar you want to modify (for example, DATAMGR.TLS for Data Manager; PLBX.TLS for Oracle Utilities Billing Component).
3. At the bottom of the file, add a line for each executable program you want to be able to launch from the menu bar.

- **For Oracle Utilities application programs, use this format:**

```
<executable-name> -u%u -p%p -c%c -f%f -q%q, &<title>
```

Example:

```
ratexprt.exe -u%u -p%p -c%c -f%f -q%q &RateExpert
```

<executable-name> is the name of the Oracle Utilities program to launch via the menu bar.

The % symbol indicates that the target program will launch using the same parameters specified for the source program (see **Oracle Utilities Application Command Line Parameters** on page 4-10 in the *Oracle Utilities Energy Information Platform Installation Guide*). These parameters are:

%u - userid

%p - password

%c - connect string

%f - config file name

%q - qualifier

<title> is the name of the program, as you want it to appear in the menu bar when the user selects the Tools option. In other words, the & symbol determines the hot key, and can be used before any characters you use to specify the name.

- **For third-party programs, use this format:**

```
<path><executable-name>, &<title>
```

Examples:

```
notepad.exe &editor
```

```
c:\program files\microsoft office\office\excel.exe, &MS Excel
```

<path><executable-name> is the absolute path and name of the program to launch via the menu bar. Programs that are in the default Windows path (such as Notepad) do not need the path specified.

<title> is the name of the program, as you want it to appear in the menu bar when the user selects the Tools option. In other words, the & symbol determines the hot key, and can be used before any characters you use to specify the name.

4. Save your changes to the file.
5. Open the Oracle Utilities application. Verify that the desired programs are available via the toolbar by selecting **Tools->[name]**. The program should open in another window.

Encrypting Configuration Files

Several Oracle Utilities configuration files contain secure information, such as the data base connection string or password. This information should be accessible only for applications or administrators.

To secure this information, the EcryptCFG.EXE console utility can be used to encrypt particular elements in a specified configuration file. Only the connection string (<CONNECTSTRING>) and/or password (<PWD> or <PASSWORD>) XML elements can be encrypted.

The syntax for this program is as follows:

EcryptCFG -f <file name> [-?]

where:

Parameter	Description
-f	Path and file name of the configuration file to be encrypted. Example: -f c:\lodestar\cfg\lssecure.cfg.xml
-?	Used to display syntax.

Chapter 3

Setting Up Energy Information Platform Database Records

This chapter describes how to set up database records used when using the Oracle Utilities Energy Information Platform, including:

- **Setting Up Energy Information Platform Lookup Tables**
- **Setting Up Other Records**

You set up database records in the Oracle Utilities Data Repository using either Data Manager or Data Navigator.

A Note about Predefined Records:

Several of the tables described in this chapter contain predefined records used by the Oracle Utilities Energy Information Platform. These records are listed in the appropriate table description. **Do not modify or delete any of these predefined records.**

Setting Up Energy Information Platform Lookup Tables

The Energy Information Platform and Oracle Utilities products use a number of common records, often referred to as Lookup tables. This section describes some of these, and how to set up records in the following tables in the Oracle Utilities Data Repository:

- **Account Statuses**
- **Bill Determinants**
- **End Use**
- **Industrial Classifications**
- **Jurisdiction**
- **LSCurrency**
- **Operating Company**
- **Revenue Code**
- **Second Per Interval**
- **Standard Industrial Classification (SIC) Code**
- **Unit of Measure (UOM)**

Account Statuses

Records in the Account Status table represent various statuses for accounts (such as Active, Inactive, etc.). Records in this table include the following information:

- **Code:** A code (up to 64 characters) identifying the account status.
- **Name:** A name for the account status code. This name should help users recognize the account status in displays and reports.

Bill Determinants

Bill determinants are measures of customer energy consumption used to compute bills. They can be metered values, or values computed by the software and written back to the database.

Records in the Bill Determinants table identify the *types* of billing determinants that can be applied. The actual customer values for the determinants are stored in the Bill History, Bill History Value, or Meter Value records. Records in this table include the following information

- **Code:** A unique code (up to 64 characters) for the billing determinant.
- **Identifier:** A unique identifier (up to 32 characters) that the Rules Language will use to refer to the billing determinant.
- **Name:** The complete name or other description (up to 64 characters) of the billing determinant. This name helps users recognize the code in displays and reports.
- **Unit of Measure:** The UOM for this billing determinant (from the **Unit of Measure (UOM)** table).
- **BILLHISTORY Column:** *View only.* If actual usage values for this billing determinant are stored in the Bill History Table, this is the name of the column in the Bill History Table where the values are stored. If the bill determinant values are stored in the Bill History Value Table, this field is blank. This field is only used if additional fields have been added to the Bill History table.
- **Aggregate:** The data aggregation method the programs will use when converting interval data using this Bill Determinant. For example, kW measured every 15 minutes should not be

added together when aggregating to hourly data, but should have an average or a maximum taken. The options are:

- **Average** - average interval values to get aggregate
- **Max** - find maximum and use as aggregate
- **Total** - add values to get aggregate.

End Use

End use is an appliance or other device whose consumption is measured for billing, end use studies, etc. End uses include: air conditioners, refrigerators, hot water heaters, furnaces, and total house. Records in this table include the following information

- **Code:** A unique code (up to 64 characters) identifying the end use.
- **Name:** The complete name or other description (up to 64 characters) of the end use. This name helps users recognize the code in displays and reports.

Industrial Classifications

Records in this table represent North American Industrial Classification Codes, which are standardized codes that are used widely to categorize organizations by their activities or products. Records in this table include the following information:

- **Code:** A unique code identifying the code.
- **Name:** An optional name for the code.

Jurisdiction

Records in the Jurisdiction table represent different geographic regions or markets in which Oracle Utilities software is being used. A fundamental assumption in the organization of the Oracle Utilities Data Repository is that accounts, rate schedules, and factors must “belong” to one operating company/jurisdiction. Records in this table include the following information:

- **Code:** A unique code (up to 64 characters) identifying the jurisdiction.
- **Name:** The complete name or other description (up to 64 characters) of the jurisdiction. This name helps users recognize the code in displays and reports.

LSCurrency

Records in the LSCurrency table represent different currency types associated with accounts in the system. LSCurrency records have the following fields:

- **Currency Code:** Indicates the three-letter ISO 4217 code to be used. If the currency symbol is not specified in the **Currency Format** column, the three-letter currency code is used to format monetary values. For example, if CURRENCYCODE is set as USD, the positive values are displayed as “12.34 USD” and negative values are displayed as “(12.34 USD)”.
- **Currency Format:** Used to format currency output values. This column contains an XML string containing the formatting information for currency. The format of the string is as follows:

```
<currency [NumDigits='Integer with max value 9'] [LeadingZero='Integer']
[Grouping='Integer'] [DecimalSep='String'] [ThousandSep='String']
[NegativeOrder='Integer'] [PositiveOrder='Integer']
[CurrencySymbol='String'] />
```

Attribute Descriptions:

- **NumDigits:** An integer that specifies the number of decimal digits to display.

- **LeadingZero:** Specifies whether or not to use leading zeros in decimal fields.
- **Grouping:** Specifies the size of each group of digits to the left of the decimal separator. Values in the range of 0 - 9 and 32 are valid. For example, grouping by thousands is indicated by 3.
- **DecimalSep:** The character used as the decimal separator.
- **ThousandSep:** The character used as the thousands separator.
- **NegativeOrder:** Specifies the negative currency mode.
- **PositiveOrder:** Specifies the positive currency mode.
- **Currency Symbol:** String indicating the currency symbol for the currency. For example, the currency symbol for US Dollars is “\$”

All attributes are optional. If an attribute is not set, the value for the attribute is taken from the Windows Regional Settings. If the Currency Symbol is not set, the Currency Code is used for formatting. In this case, Positive/Negative order parameters set will not have any effect.

- **Is Default:** A flag that indicates if the currency is the default for the database. If there are more than one record in the LSCurrency table, one must be set as the default.

Operating Company

Records in the Operating Company table represent business entities that use Oracle Utilities software to price and/or bills accounts, perform settlement calculations, and other functions. Operating company codes are referred to in records throughout the Oracle Utilities database. Records in this table include the following information:

- **Code:** A unique code (up to 64 characters) identifying the operating company.
- **Name:** The complete name or other description (up to 64 characters) of the operating company. This name helps users recognize the code in displays and reports.
- **Address, City, State, Zip Code, and Phone:** *Optional.* This information is used primarily by Oracle Utilities Billing Component’s Bill Fax/E-Mail feature, which prints it as the return address on billing information output in fax- or e-mail-ready format. See the *Billing Expert User’s Guide* for details.
- **Directory - Contact:** The directory record (from the **Directory** table that corresponds to the operating company).

Revenue Code

Records in this table identify a class of accounts (for example, Commercial or Residential). Records in this table include the following information:

- **Code:** A unique code (up to 64 characters) identifying the business class.
- **Name:** A descriptive name (up to 64 characters) for the business class.

Second Per Interval

Records in the Second Per Interval table define specific intervals (in seconds per interval) in which interval data is recorded. Records in this table include the following information:

- **SPI:** The number of seconds per interval (60, 300, 900, 1800, 3600, etc.).
- **Description:** A description of the interval.

This table is populated with the following pre-defined records:

SPI	Description
60	1 minute intervals
300	5 minute intervals
900	15 minute intervals
1800	30 minute intervals
3600	1 hour intervals
86400	1 day intervals

Note: The above predefined records are used by the Oracle Utilities Energy Information Platform, and **should not be modified or deleted**.

Standard Industrial Classification (SIC) Code

Records in this table represent Standard Industrial Codes (SICs), which are standardized, 1- to 6-digit codes that are used widely to categorize organizations by their activities or products. These records are typically pre-loaded. Records in this table include the following information:

- **Code:** A unique code (up to 6 characters) identifying the category.
- **Note:** An optional note, up to 254 characters.

Unit of Measure (UOM)

A Unit of Measure code is an Oracle Utilities or utility-specified code that identifies measured quantities of interest, such as KWH, KWH-OUT, LEADING KVARH, LAGGING KVARH, and temperature. The physical units measured (kWh, kvarh, etc.) are referred to as “units.”

Records in this table include the following information:

- **Code:** A unique code (up to 64 characters) identifying the unit of measure. Codes 01 through 99 are reserved for Oracle Utilities UOM codes. **Do not modify or delete any of these predefined records.** Changes made to these records can result in errors when using Oracle Utilities applications.
- **Name:** The complete name or other description for the UOM (up to 64 characters). This name helps users recognize the code in displays and reports.
- **Unit:** The physical unit measured; for example, the physical unit for the UOMs KWH-IN and KWH-OUT is KWH. Some Oracle Utilities programs totalize channels with the same UOM; others totalize channels with the same units.
- **Aggregate Method:** The method the programs will use to aggregate data when converting interval data in this UOM to a lower frequency. For example, kW measured every 15 minutes should *not* be added together when aggregating to hourly data, but should have an average or a maximum taken. Select from three options:
 - **Average** - average interval values to get aggregate
 - **Max** - find maximum and use as aggregate
 - **Total** - add values to get aggregate.
- **Energy to Demand?** A flag that indicates if the data in this UOM can be converted to demand, Yes or No? If set to Yes, quantities assigned this UOM are available to functions that convert energy to demand using the formula: Demand = Energy x Intervals per Hour.

- **Totalize Method:** Specifies how the programs will treat values for intervals or determinants to arrive at a total when combining data from two or more recorder channels, or when creating bill frequency tables:
 - **Average** - average values to get total
 - **Max** - find maximum to get total
 - **Total** - add values to get total.

Note: If the DO_NOT_USE_UOM_FOR_SCALING parameter is included in the LODESTAR.CFG file, intervals and determinants are totalized regardless of this setting.

Overriding and/or Extending the Default UOM List

The Oracle Utilities Energy Information Platform comes with pre-defined list of units-of-measure, which represent the “default” list of UOMs used by Oracle Utilities applications. It is possible to extend or override this “default” list if needed through the INTDCONFIG.CFG.XML configuration file. This file can specify a UOM source that will override and/or extend the default list. This source can be loaded either from a data source (specifically the Unit-of-Measure table in the data source) or a list of UOMs specified in the configuration file. The Oracle Utilities applications load the default UOM list first, and then override any duplicated UOMs or add new UOMs defined in the UOM source that are not in the default list. See

INTDCONFIG.CFG.XML on page 2-17 for more information.

Setting Up Other Records

In addition to lookup records, the Energy Information Platform and other Oracle Utilities applications use other common records. This section describes these and how to set up records in the following tables in the Oracle Utilities Data Repository.

- **Address**
- **Address Format**
- **Business Day**
- **Business Calendar**
- **CIS Account**
- **Contact Method Type**
- **Contact Purpose**
- **Directory**
- **Factor**
- **Factor Values**
- **Group Category**
- **Interrogations**
- **Interval Data Sources**
- **LS Address**
- **LS Market**
- **LS Premise**
- **Market Attribute**
- **Market Attribute Type**
- **Market Participant Relationship Type**
- **Market Participant Type**
- **MV90 Status Code Map**
- **Organization**
- **Override**
- **Participant Type**
- **Web Service**
- **Web Service Type**

Address

Records in the Address table represent individual addresses that can be associated with customer, accounts, and other entities in the system. Records in this table include the following information:

- **Address 1:** Street address.
- **Address 2:** Additional street address information (such as postal box)
- **Address 3:** Additional street address information (such as postal box)
- **City:** City associated with the address.
- **State:** State associated with the address.
- **Postal Code:** Postal code associated with the address.
- **County:** County associated with the address.
- **Country:** Country associated with the address.

Address Format

Records in the Address Format table represent specific address formats available for addresses stored in the LS Address table (used with Service Points and Market Participants). This allows locale-specific address formats to be configured and accessed via contact screens in the Energy Information Platform user interface. Records in this table include the following information:

- **Address Format:** The name of the address format.
- **VAL1NAME:** The name of the first value in the address format.
- **VAL2NAME:** *Optional.* The name of the second value in the address format.
- **VAL3NAME:** *Optional.* The name of the third value in the address format.
- **VAL4NAME:** *Optional.* The name of the fourth value in the address format.
- **VAL5NAME:** *Optional.* The name of the fifth value in the address format.
- **VAL6NAME:** *Optional.* The name of the sixth value in the address format.
- **VAL7NAME:** *Optional.* The name of the seventh value in the address format.
- **VAL8NAME:** *Optional.* The name of the eighth value in the address format.

The Address Format table contains the following predefined record:

Column	Value
Address Format	US Format
VAL1NAME	Address 1
VAL2NAME	Address 2
VAL3NAME	Address 3
VAL4NAME	City
VAL5NAME	State
VAL6NAME	ZIP Code
VAL7NAME	Country

Business Calendar

Records in the Business Calendar table represent business calendars used by the Scheduler service and the Oracle Utilities Billing Component - Workflow Management application. Records in this table include the following information:

- **Operating Company Code:** The operating company that uses this business calendar.
- **Jurisdiction Code:** The jurisdiction that uses this business calendar.

Note: If you want the business calendar to be “global”—that is, available across all operating companies and jurisdictions—leave the operating company and jurisdiction Null.

- **Business Calendar Name:** Name of the business calendar.

Business Day

Records in the business day calendar represent individual business days associated to a specific business calendar. Records in this table include the following information:

- **Business Calendar:** Date and time the CIS account became active.
- **Date:** The date for the business day.
- **Valid Shutoff Day:** A flag that indicates if the business day is a valid day for shutting off service to an account (used with Oracle Utilities Receivables Component and Collections).

CIS Account

The hierarchy of entities recognized by the Oracle Utilities products may not match those recognized by a Customer Information System (CIS). For example, an account and its service locations may each have a unique account ID in a CIS; however, in the Oracle Utilities Data Repository, only the account should have an account ID. The CIS Account Table makes it possible to “map” the levels of entities recognized by the Oracle Utilities products to the IDs recognized by a CIS. Records in this table include the following information:

- **ID:** The ID recognized by your CIS.
- **Start Time:** Date and time the CIS account became active.
- **Stop Time:** Date and time the CIS account became inactive (Null if still active).
- **Name:** Name of the CIS Account.
- **Note:** Optional descriptive information.

Contact Method Type

Records in the Contact Method Type table represent specific types of contact methods that can be associated to contacts, such as email addresses, phone numbers, etc. Records in this table include the following information:

- **Type:** The type of contact method (email, phone, cell, fax, web address, etc.)
- **Subtype:** An optional subtype (work, home, etc.) for the contact method type.

The Contact Method Type table contains the following predefined records:

Type	Subtype
Cell	Home
Email	Home

Type	Subtype
Fax	Home
Pager	Home
Phone	Home
Web Address	Home
Cell	Work
Email	Work
Fax	Work
Pager	Work
Phone	Work
Web Address	Work

Contact Purpose

Records in the Contact Purpose table represent purposes for contacts (billing, general business, etc.) used with contacts in the Energy Information Platform. Records in this table include the following information:

- **Contact Purpose:** the contact purpose.

Directory

Records in the Directory table represent contacts associated with customers and/or accounts. This table is also used by Oracle Utilities Billing Component's Bill Fax/E-Mail feature (see the *Oracle Utilities Billing Component User's Guide* for complete details about setting up this feature). You can think of the Directory as a rolodex containing the names of potential recipients of the faxes or e-mail, as well as the names of utility personnel that the customer can contact with questions. Records in this table include the following information:

- **Directory Code:** A unique code (up to 64 characters) identifying this record.
- **Name:** The name of the account manager, utility contact, or other recipient.
- **Phone #1:** The phone number of the account manager, utility contact, etc.
- **Phone #2:** An alternate phone number.
- **Fax:** The Fax number for the account manager, utility contact, etc.
- **Pager:** The Pager number of the account manager, utility contact, etc.
- **Email:** The e-mail address of the recipient (account manager, utility contact, etc.).
- **Title:** The title of the account manager, utility contact, or other recipient.
- **Name of File:** The name of the E-mail file, without the extension. Oracle Utilities Billing Component writes the file to the LODESTAR\USER directory, and adds the .txt extension. If you do not supply a name, Oracle Utilities Billing Component uses the default (EMAIL.TXT).

Note: It is strongly recommended that you associate a unique file name with each Directory record (the name in the second field, for example). If you use the same file name for multiple records, Oracle Utilities Billing Component will append each new e-mail to the end of the file.

Note about faxes: If you are creating a fax, Oracle Utilities Billing Component ignores any file name you supply here and uses the default (FAX.TXT).

Factor

Records in the Factor table represent variables that can be used in Oracle Utilities Rules Language rate forms to represent charges, taxes, and other values that are used widely and are expected to change over time. The factor names are kept in the Factor Table, and the actual values associated with the factor are kept in the **Factor Values** Table. When a factor changes, you can create a new Factor Value record, instead of changing the value in every rate form, and the programs automatically find and use the correct value. Records in this table include the following information:

- **Operating Company Code:** The operating company that uses this factor.
- **Jurisdiction Code:** The jurisdiction under which this factor is applied.

Note: If you want the factor to be a “global” factor—that is, available across all operating companies and jurisdictions—leave the Null checkmarks in place for operating company and jurisdiction. For example, you would probably set up “yearly interest rate” as a global factor, because it is not specific to an operating company/jurisdiction.
- **Code:** A code (up to 64 characters) identifying the factor. This code is used in the rate form scripts to apply the factor.

Note: The combination of Operating Company/Jurisdiction/Code must be unique.
- **Name:** A name for the factor. This name should be unique so that users are not confused by multiple factors with the same name, and should help users recognize the factor in displays and reports.
- **Unit of Measure:** The UOM for the quantity represented by the factor value.

Factor Values

Factor value records are child records of a Factor record, and store the actual values and prorate methods associated with each Factor.

Note: Whenever a value changes for a factor, you must add a new Factor Value record. This preserves historical versions.

The first three fields in this table are automatically filled in using information from the parent Factor record. The three remaining fields are:

- **Effective Date:** The date that the factor went into effect, using the YYYY-MM-DD HH:MM:SS format.
- **Value:** The value of the factor.
- **Prorate?:** A flag that indicates whether the programs prorate charges if the factor value changes during a bill period:
 - **Yes** - Prorate the charges.
 - **No** - Do not prorate the charges (apply the factor value that was in effect on the Bill Date to the entire bill period).

Group Category

Records in the Group Category table represent groupings or subsets of interval data recorder,channel pairs. This allows you to form subsets of an account's recorder channels (e.g., assigning channels 1 and 2 to a subset called "Group 1," and assigning channels 3 through 5 to "Group 2"), which in turn allows you to keep track of each group's usage values separately. You can also group a single account's channels differently for different categories of applications, such as billing, load research, and so on. Records in this table include the following information:

- **Code:** A unique code (up to 64 characters) identifying the group category.
- **Note:** An optional note, up to 254 characters.

Interrogations

Records in the Interrogations table specify a billing "eligibility" window for types of interval data recorders. The types are based on how data is retrieved from them.

Note: See the *Oracle Utilities Billing Component User's Guide* for a definition of "eligibility" in the billing process.

If the account has pre- and post-windows defined in the Account History Table, those values take precedence over this setting. Records in this table include the following information:

- **Code:** A unique code (up to 64 characters) identifying the interrogation category.
- **Name:** The complete name or other description (up to 64 characters) of the interrogation category. This name helps users recognize the code in displays and reports.
- **Prewindow:** The number of days before the account's scheduled read date that Oracle Utilities Billing Component's Automatic Billing module begins scanning for billing data for this recorder type. Oracle Utilities Billing Component continues scanning until the data becomes available, or until the post-window closes. You can supply any value from 0 to 28.
- **Postwindow:** The number of days after the account's scheduled read date that the Automatic Billing module continues to scan for billing data for this recorder. You can supply any value from 0 to 28. If you set the post-window to 0, the data must be available by the end of the read date or it is considered missing.

Interval Data Sources

Records in the Interval Data Sources table represent individual interval data sources available to the Energy Information Platform. Records in this table include the following information:

- **Name:** A unique name for the interval data source.
- **Type:** The type of interval data source. This can be one of the following:
 - **Relational:** Indicates that the data source is a set of relational tables in the Oracle Utilities Data Repository.
 - **BTE:** Indicates that the data source is a *.BTE file.
- **Location:** The location of the data source.
 - **When Type = Relational:** The name of the table that contains the header information for the interval data contained in the data source. For most sources this will be "LSCHANNELCUTHEADER."
 - **When Type = BTE:** The fully qualified path and file name or UNC to the *.BTE file. The filename cannot have spaces. Note that this path should be from the web server to the file.

Note: When accessing interval data in BTE files, Pervasive.SQL server software must be installed on the machine where the BTE is located, and Pervasive.SQL client (or server) software must be installed on the web server.

- **Read Only:** A flag that indicates (True or False) if the data source can be edited using the Interval Data Manager component of the Energy Information Platform.
- **Audit:** A flag that indicates (True or False) if archive records are created when interval data in the data source is edited using the Interval Data Manager component of the Energy Information Platform. Cuts in data sources with the Audit flag set to “False” cannot be restored using the Restore Archive Interval Data Manager function.
- **Version Channel Cut Header:** the name of the Interval Data Version table in which versioned cuts from the data source are stored. The Version Channel Cut Header table associated with the Channel Cut Header table is the LSCHVERSION (Channel Cut Header Version) table.

Note: This field applies only to interval data sources whose type is **Relational**. Also, the Audit flag MUST be FALSE if a table is specified here.

- **Load Analysis Database Type:** the type of Load Analysis database (ALDB, CLDB, ELDB, or SLDB). The default setting is blank, which means the interval datasource is not available for Load Analysis.

Interval Data Manager and Oracle Utilities Load Analysis Data Sources

When creating interval data sources for use with Interval Data Manager and the Oracle Utilities Load Analysis load research tool in the Interval Data Sources table, note the following:

- The “Audit” field for ELDB and SLDB data sources should be set to “FALSE.”
- GLDB and RLDB data sources should NOT be made available to Interval Data Manager.

LS Address

Records in the LS Address table represent individual addresses that can be associated with service points, market participants, contacts, and other entities in the system. Records in this table include the following information:

- **Address Format:** The name of the address format that the address is based upon, from the **Address Format** table.
- **VAL1:** The first value in the address format.
- **VAL2:** *Optional.* The second value in the address.
- **VAL3E:** *Optional.* The third value in the address.
- **VAL4:** *Optional.* The fourth value in the address.
- **VAL5:** *Optional.* The fifth value in the address.
- **VAL6:** *Optional.* The sixth value in the address.
- **VAL7:** *Optional.* The seventh value in the address.
- **VAL8:** *Optional.* The eighth value in the address.

LS Market

Records in the LS Market table represent markets related to service points, market participants and other entities. Most often, records in this table also represent the markets in which the Oracle Utilities software is used. Records in this table include the following information:

- **Market ID:** Identifier for the market.

- **Service Type:** *Optional.* The service type associated with the market, from the Service Type table.

LS Premise

Records in the LS Premise table represent premises related to service points, market participants and other entities. Records in this table include the following information:

- **Premise ID:** Identifier for the premise.
- **Address:** The address of the premise, from the **LS Address** table.

Market Attribute

Records in the Market Attribute table represent market attributes that can be related to service points. Records in this table include the following information:

- **Market Attribute Type:** The market attribute type associated with the attribute, from the **Market Attribute Type** table.
- **Attribute Code:** A code that designates the market attribute.
- **Attribute Name:** The name of the market attribute.

Market Attribute Type

Records in the Market Attribute Type table represent types of market attributes that can be related to service points. Records in this table include the following information:

- **Market ID:** The market to which the attribute is applicable, from the **LS Market** table.
- **Attribute Type:** The name of the attribute type.
- **Value Type:** The type of value for the attribute type. Can be one of the following:
 - Lookup
 - Float
 - DateTime
 - String
 - Boolean

Market Participant Relationship Type

Records in the Market Participant Relationship Type table represent types of relationships between market participants. Records in this table include the following information:

- **Relationship Type:** The name of the relationship type.
- **Relationship Type From:** The “from” party in the relationship type. For example, if the record is defining the relationship from a corporate office to a factory, this column would be set to “Corporate.”
- **Relationship Type To:** The “to” party in the relationship type. For example, if the record is defining the relationship from a corporate office to a factory, this column would be set to “Factory.”
- **Description:** A description of the relationship type.

Market Participant Type

Records in the Market Participant Type table represent types of market participants. Records in this table include the following information:

- **Market ID:** The market in which the market participant participates, from the **LS Market** table.
- **Participant Type:** The type of participant, from the **Participant Type** table.

MV90 Status Code Map

Records in the MV90 Status Code Map table are used to map MV-90 status codes to LODESTAR status codes on import of MV90 data. Records in this table include the following information:

- **MV90 Status Word:** The MV90 status word. Used to define the type of MV90 status code. Valid values include:
 - Channel Status: Defines a mapping for MV90 Channel Status codes
 - Interval Status: Defines a mapping for MV90 Interval Status codes
- **Bit:** The bit associated with the status code.
- **LODESTAR Status Code:** The LODESTAR status code that the MV90 code is mapped to.
- **Priority:** A priority associated with the status code. Priority levels are represented as integers, starting at “1” as the lowest priority. The higher the number - the higher the priority.
- **Description:** A description of the status code.

Organization

Records in the Organization table represent organizations related to market participants. Records in this table include the following information:

- **Organization Name:** The name of the organization.
- **Address:** The address of the organization, from the **LS Address** table.

Override

Under special circumstances, such as interruptions and backup periods, or by other contractual arrangement such as an economic development discount, an account may be subject to charges that differ from those that normally apply. Overrides are a mechanism that make it possible to apply these special charges, and to keep track of when they were applied.

Note: Here, you are simply assigning a lookup code to each possible override in the system. Assigning these overrides to accounts is explained in the *Oracle Utilities Billing Component installation and Configuration Guide*.

Records in this table include the following information:

- **Code:** A unique code (up to 64 characters) identifying the override.
- **Name:** A unique name for the override. This name will be used in the rate forms to apply the override.

Participant Type

Records in the Participant Type table represent types of participants (commercial, residential, etc.). Records in this table include the following information:

- **Participant Type:** The name of the participant type.
- **Description:** A description of the participant type.

Web Service

Records in the Web Service table represent individual web services used by the Adapter. Records in this table include the following information:

- **Web Service ID:** The ID of the web service. This is used when calling the web service.
- **Description:** A description of the web service
- **Web Service Type:** The type of web service, from the Web Service Type table
- **Runtime Service:** An optional Runtime Service (from the Runtime Service table) associated with the web service
- **Business Rule:** An optional Business Rule (from the Business Rule table) associated with the web service

Web Service Type

Records in the Web Service Type table represent different types of web services available via the Adapter. Records in this table include the following information:

- **Web Service Type Code:** A code that designates the web service type
- **Description:** A description of the web service type

The Web Service Type table includes the following predefined records:

Web Service Type Code	Description
ABR	Adapter Business Rule
ARS	Adapter Runtime Service

Chapter 4

Configuring Interval Data Viewer and Interval Data Manager

This chapter describes how to configure the Interval Data Viewer and Interval Data Manager components of the Oracle Utilities Energy Information Platform, including:

- **Configuring Interval Data Viewer**
 - **Prepare Interval Data**
 - **Set Up Interval Data Sources**
 - **Assign Data and Feature Privileges**
- **Configuring Interval Data Manager**
 - **Prepare Interval Data**
 - **Set Up Interval Data Sources**
 - **Assign Data and Feature Privileges**

Configuring Interval Data Viewer

This section describes how to set up and configure Interval Data Viewer, including:

- **Prepare Interval Data**
- **Set Up Interval Data Sources**
- **Assign Data and Feature Privileges**

Prepare Interval Data

Preparing interval data involves importing interval data into the interval data tables in the Oracle Utilities Data Repository. Interval data tables are sets of tables used to store interval data in a relational format, and include the following:

For standard interval data:

- A header table (stores header information for each cut)
- A data table (stores interval data values and status codes for each cut)
- A validation messages table (stores validation messages for each cut, used with standard interval data tables)
- An edit trails table (stores edit trails for each cut)

For enhanced/generic interval data:

- A parent table (stores parent records)
- A header/data table (stores header information, as well as interval data values and status codes for each cut)
- An edit trails table (stores edit trails for each cut, optional)

The Channel Cut Header table (LSCHANNELCUTHEADER), Channel Cut Data table (LSCHANNELCUTDATA), Channel Cut Validation Message table (LSCHANNELCUVMSG), and Channel Cut Edit Trails table (LSCEDITRAILS) are the default standard interval data tables supplied with the base database schema, but additional interval data table sets may be used as well.

Interval data can be imported using Data Manager or the INTDIMP.EXE command line program. See **Chapter 3: Importing Data** in the *Data Manager User's Guide* for more information about importing interval data. You can also import interval data using the Import Data or Upload Data features in the Oracle Utilities Energy Information Platform user interface. See **Import Data** on page 6-25 or **Upload Data** on page 6-27 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about importing interval data using these features.

Note: Interval Data Viewer does not support interval data in Btrieve (Pervasive.SQL) files.

Set Up Interval Data Sources

If interval data is stored in more than one interval data table set, you should also create records in the Interval Data Sources table for each. See **Interval Data Sources** on page 3-12 for more information about this table.

Assign Data and Feature Privileges

Before users can view interval data using Interval Data Viewer, you must assign the appropriate feature and data privileges to the users and groups of users who will use this feature. See **Entity Management Features** on page 7-5 for more information about granting feature privileges related to Interval Data Viewer. See **Adapter Features** on page 7-13 for more information about granting data privileges related to Interval Data Viewer.

Configuring Interval Data Manager

This section describes how to set up and configure Interval Data Manager, including:

- **Prepare Interval Data**
- **Set Up Interval Data Sources**
- **Assign Data and Feature Privileges**
- **Set Up Validation Environment File (Optional)**

Prepare Interval Data

Interval Data Manager can access interval data stored in both the Oracle Utilities Data Repository as well as in Btrieve (Pervasive.SQL) files.

Preparing Interval Data In Btrieve Files

Preparing interval data in Btrieve files involves making sure that the files are accessible by the web server, and making sure that the Pervasive.SQL software has been installed on the web server. The network path from the web server to the server on which the Btrieve files are stored will be needed when creating records in the Interval Data Sources table (see **Set Up Interval Data Sources** on page 4-4).

Preparing Interval Data in the Oracle Utilities Data Repository

Preparing interval data in the Oracle Utilities Data Repository involves importing interval data into the interval data tables in the Oracle Utilities Data Repository. Interval data tables are sets of tables used to store interval data in a relational format, and include the following:

For standard interval data:

- A header table (stores header information for each cut)
- A data table (stores interval data values and status codes for each cut)
- A validation messages table (stores validation messages for each cut, used with standard interval data tables)
- An edit trails table (stores edit trails for each cut)

For enhanced/generic interval data:

- A parent table (stores parent records)
- A header/data table (stores header information, as well as interval data values and status codes for each cut)
- An edit trails table (stores edit trails for each cut, optional)

The Channel Cut Header table (LSCHANNELCUTHEADER), Channel Cut Data table (LSCHANNELCUTDATA), Channel Cut Validation Message table (LSCHANNELCUVMSG), and Channel Cut Edit Trails table (LSCEDITRAILS) are the default standard interval data tables supplied with the base database schema, but additional interval data table sets may be used as well.

Interval data can be imported using Data Manager or the INTDIMP.EXE command line program. See **Chapter 3: Importing Data** in the *Data Manager User's Guide* for more information about importing interval data. You can also import interval data using the Import Data or Upload Data features in the Oracle Utilities Energy Information Platform user interface. See **Import Data** on page 6-25 or **Upload Data** on page 6-27 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about importing interval data using these features.

Set Up Interval Data Sources

You must also create records in the Interval Data Sources table for each source of interval data that Interval Data Manager will access, including both relational interval data table sets as well as Btrieve (Pervasive.SQL) files. See **Interval Data Sources** on page 3-12 for more information about this table.

Assign Data and Feature Privileges

Before users can access and edit interval data using Interval Data Manager, you must assign the appropriate feature and data privileges to the users and groups of users who will use this feature. See **Common Features** on page 7-8 for more information about granting feature privileges related to Interval Data Manager. See **Adapter Features** on page 7-13 for more information about granting data privileges related to Interval Data Manager.

Set Up Validation Environment File (Optional)

Interval Data Manager can be configured to support automatic validation of interval data cuts upon saving. When validation is enabled, cuts are automatically validated via the Oracle Utilities Load Analysis Load Data Management System Cut Series Validation program prior to being saved.

To enable automatic validation, you must create an environment file that defines the specifics concerning the validations to be performed. This file must be named **TGX21B.ENV**, and must be put into the **C:\LODESTAR\CFG** directory on the web server.

See **Chapter 5: Setting Up Your Oracle Utilities Load Analysis System** in the *Oracle Utilities Load Analysis Load Data Management User's Guide* for more information about creating this file and available validations options.

Chapter 5

Configuring Work Queues

This chapter describes how to configure Work Queues, including:

- **Work Queues Tables Overview**
- **Work Queues Actions**
- **Defining Your Work Queues Configuration**
- **Configuring Work Queues and Work Queue Types**
 - **Setting Up Lookup Tables**
 - **Setting Up Work Queue Types**
- **Creating Work Queue Items**
 - **Using Oracle Utilities Billing Component**
 - **Using the Oracle Utilities Rules Language**
 - **Using External Applications**

Work Queues Tables Overview

Understanding how the work queues data is stored and how items are acted upon is important when configuring the Work Queues application. The Work Queues application uses the following tables in the Oracle Utilities Data Repository.

- **Business Process:** Records in this table represent business processes which can be used in defining a work queue type.
- **LODESTAR Product:** Records in this table represent specific Oracle Utilities products which can be used in defining a work queue type.
- **Work Queue:** Records in this table represent queues into which work queue items are grouped.
- **Work Queue Type:** Records in this table represent types of work queue items available.
- **Work Queue Approval Level:** Records in this table represent different levels of approval needed when approving work queue items.
- **Work Queue Approval Procedure:** Records in this table represent approval procedures used by work queue item types.
- **Work Queue Approval Step:** Records in this table represent individual steps associated with an approval procedure and work queue item type.
- **Work Queue Open Action:** A history of all actions performed on each item is maintained in the database. Action records will be created automatically for any activity that updates the state of an item. Records in this table represent individual actions performed on open work queue items. See **Work Queues Actions** on page 5-3 for more information about work queue functions.
- **Work Queue Open Item:** Records in this table represent open work queue items. These are items currently being reviewed and worked on.
- **Work Queue Closed Action:** A history of all actions performed on each item is maintained in the database. Action records will be created automatically for any activity that updates the state of an item. Records in this table represent individual actions performed on closed work queue items. See **Work Queues Actions** on page 5-3 for more information about work queue functions.
- **Work Queue Closed Item:** Records in this table represent closed work queue items. These are items that have been closed.
- **Work Queue Customer Parameter:** Records in this table represent custom parameters used by one or more work queue item types.
- **Work Queue Type Parameter:** Records in this table relate custom parameters to specific work queue item types.
- **Work Queue Hint:** Records in this table represent hints available to work queue item types.
- **Work Queue Type Hint:** Records in this table relate hints to specific work queue item types.

See **Configuring Work Queues and Work Queue Types** on page 5-11 for information about creating records in these and other tables.

Work Queues Actions

This section describes the actions that can be performed on work queue items.

Opening a New Item

The **Open** action opens new work queue items. Work queue items can be opened via the Work Queues COM interface, the Oracle Utilities Rules Language, or from the Oracle Utilities Billing Component application. See **Creating Work Queue Items** on page 5-20 for more information about opening work queue items.

Action records are not created when opening an item.

If the item is “Approval Only” and an approval process is associated with it, then the Approval Step Number and Approval Level attributes will be set as defined in the first step of that approval process. If the item is “Approval Only”, then it will effectively be in an “awaiting approval” state at this time.

Listing Items

The web-enabled Work Queues user interface allows both administrators and users to search for and list existing work queue items. A user's security permissions and areas will be used to filter which items the user is allowed to view. A number of additional sorting and filtering parameters will be available as detailed below.

Sorting

Records are filtered by a user-selected value, with the following fields used as secondary/tertiary sorts:

- Work Queue Type
- Work Queue

User can also sort on any one the following attributes:

- Work Queue Type
- UID (unique id of the work queue item in the system)
- Work Queue
- Priority Level
- Work by Time
- Opened Time
- Opened Note
- Assigned to User
- Approval Level
- Approval Only flag
- Approved flag
- LODESTAR Product
- Business Process

In addition, users can sort on any custom parameters associated with the work queue item types being viewed.

Filtering

In addition to the filtering that automatically occurs based on a user's security permissions and areas, user's can also explicitly filter on the above attributes.

Assigning an Item

The **Assign** action assigns an item (or items) to a user. This action requires the item id, the assigned to user, the performed by user, and an optional note. This action updates the Assigned To User and Assigned To Time attributes of the item, and creates a new action record for the item. The action type is "Assigned" and the action number is the next consecutive one for the item. The Details attribute indicates to whom the item is being assigned and, if a reassignment, who the item was previously assigned to.

Only open items can be assigned. Permission must be granted to users in order to assign items.

Assigning items can be performed using the Work Queues user interface and the Work Queues COM interface.

Unassigning an Item

The **Unassign** action unassigns an item (or items) from its previously assigned user. This action updates the Assigned to User and Assigned To Time values on a work queue item to NULL, and creates an action for the item. The action type is "Unassigned" and the action number is the next consecutive one for the item. The Details attribute indicates what user the item was previously assigned to, the Performed By User and an optional Note.

Only open and assigned items are allowed to be unassigned. Permission must be granted to users in order to unassign items.

Unassigning items can be performed using the Work Queues user interface and the Work Queues COM interface.

Updating an Item

The **Update** action updates the attributes of an item or items. The following attributes may be explicitly updated for an item:

- Queue
- Priority Level
- Work By Time
- Data
- (Note Only)

This action updates the specified attribute and creates an action for the item. The action type is “Updated” and the action number is the next consecutive one for the item. The Details attribute indicates what attribute was updated as well as what its previous value was. The action also indicates the Performed By User and an optional Note (this action may be used to add a note only to the item).

Only open items may be updated. Permission must be granted to users in order to update items.

Updating items can be performed using the Work Queues user interface and the Work Queues COM interface. Updating items using the Work Queues COM interface allows updating of times one at a time.

Resolving an Item

The **Resolve** action resolves an item (or items) when the item has been completed to satisfaction. This action updates the Assigned To User and Assigned To Time values of the item to NULL and creates an action for the item. The action type is “Resolved” and the action number is the next consecutive one for the item. The action also indicates the Performed By User and an optional Note. If the current Assigned To User is the same as the Performed By User, the Work Time Span is set to the difference between the Action Time and the current Assigned To Time (displayed in Days, Hours, and Minutes).

Depending on the state of the item, this action also does one of the following:

1. If the item does not have an associated Approval Procedure, the Close action, as described below, is invoked.
2. If the item has an associated Approval Procedure, the Approval Step Number and Approval Level attributes of the item are updated as defined by the first step of that approval process. The Approval Step Count is also updated to zero, and the Work By Time is calculated from the Approve By Hours values of the associated type. At this point, the item is effectively in an “awaiting approval” mode.

Only open, unresolved, non-approval only items may be resolved. Permission must be granted to users in order to resolve items.

Resolving items can be performed using the Work Queues user interface and the Work Queues COM interface.

Approving an Item

The **Approve** action approves an item (or items) if it is in an “awaiting approval” mode. This action updates the Assigned To User and Assigned To Time values of the item to NULL and creates an action for the item. The action type is “Approved” and the action number is the next consecutive one for the item. The Details attribute indicates the current approval level, if one exists. The action also indicates the Performed By User and an optional Note. If the current Assigned To User is the same as the Performed By User, the Work Time Span should be set to the difference between the Action Time and the current Assigned To Time, in hours.

Depending on the state of the item, this action also does the following:

1. If the item has a current Approval Step associated with it, the action increments the current Approval Step count and compares it with the required number of approvals for that step as defined by its Approval Procedure. If the new count is greater than or equal to the required number of approvals, the Approval Step number and Level are updated to the next step as defined by the Approval Procedure (or NULL if there is no next step), the Approval Step count should be reset to zero (or NULL if there is no next step), and the Work By Time should be calculated from the Approve By Hours values of the associated type (or NULL if there is no next step).
2. If, after step 1 is complete, the item does not have a current Approval Step associated with it, the Close action is invoked. Also, if the item is “Approval Only”, the Approved attribute is updated to “Y” (true).

Only open, “awaiting approval” items may be approved. The same user may not approve the same item within the same approval step more than once. Permission must be granted to users in order to approve items.

Approving items can be performed using the Work Queues user interface and the Work Queues COM interface.

Rejecting an Item

The **Reject** action rejects an item (or items) if it is in an “awaiting approval” mode. This action updates the Assigned To User and Assigned To Time values of the item to NULL and creates an action for the item. The action type is “Rejected” and the action number is the next consecutive one for the item. The Details attribute indicates the current approval level, if one exists. The action also indicates the Performed By User and an optional Note. If the current Assigned To User is the same as the Performed By User, the Work Time Span is set to the difference between the Action Time and the current Assigned To Time, in hours.

Depending on the state of the item, this action also does the following:

1. If the item has a current Approval Step associated with it, the action updates the Approval Step, Level, and Step Count to NULL.
2. If the item is “Approval Only”, the Close action is invoked.
3. If the item is not “Approval Only”, the Work By Time is updated based on the Default Work By Hours of the associated type and the original Opened Time.

Only open, “awaiting approval” items may be rejected. Permission must be granted to users in order to reject items.

Rejecting items can be performed using the Work Queues user interface and the Work Queues COM interface.

Closing an Item

The **Close** action closes an open item (or items). An item may either be explicitly forced closed, or closed as a result of Resolving, Approving, or Rejecting the item. This action deletes the item from the **Work Queue Open Item** table and inserts it into the **Work Queue Closed Item** table. The Closed Time is set to the current time, and an additional action record is created. The action type will be “Closed” and the action number is the next consecutive one for the item. The action also indicates the Performed By User and an optional Note. If the current Assigned To User is the same as the Performed By User, the Work Time Span should be set to the difference between the Action Time and the current Assigned To Time, in hours.

Work Queue Types can be configured to invoke a specified COM method upon the closing of a work queue item of that type. See **Close Item Callback** on page 5-15 for more information.

Only open items may be closed. Permission must be granted to users in order to close items.

Closing items can be performed using the Work Queues user interface and the Work Queues COM interface.

Reopening an Item

The **Reopen** action opens a previously closed item (or items). This action deletes the item from the **Work Queue Closed Item** table and inserts it into the **Work Queue Open Item** table. The Work By Time is either explicitly provided or recalculated based on the associated work queue type's Default Work By Time and the current time. An action record is created for the item. The action type is “Reopened” and the action number is the next consecutive one for the item. The action also indicates the Performed By User and an optional Note.

Only closed items may be reopened. Permission must be granted to users in order to reopen items. Reopening items can be performed using the Work Queues user interface and the Work Queues COM interface.

Defining Your Work Queues Configuration

Before you start to setup work queues and work queue types, you need to design and define your work queues configuration. This involves determining the specific work queues and work queue types you need to support your work queue requirements, and defining those queues and types in terms usable by the Work Queues application. Defining your work queues configuration involves the following steps:

- **Defining Work Queues**
- **Defining Work Queue Types**

Defining Work Queues

The first step is to describe and define the work queues needed for your configuration. Work queues can be thought of as “buckets” into which work items are grouped. You can create work queues based on any number of factors, such as specific products and/or business processes that will use the Work Queues application, or the number of users who will work with work queue items.

Defining Work Queue Types

The next step is to describe and define the specific types of work queue items needed for your work queues configuration. You should create a work queue type for every type of exception, message, or action item that you want to use with the Work Queues application.

For example, if you’re using the Work Queues application with Oracle Utilities Billing Component, you should create work queue types for each type of billing exception or problem that might occur (these work queue types take the place of Account Notes in Oracle Utilities Billing Component). You might even create work queue types that apply to specific functions within Oracle Utilities Billing Component, such as Automatic or Approval Required Billing. If you’re using the Work Queues application with Oracle Utilities Load Profiling and Settlement, you might set up work queue types for each process performed by the application, such as profile application, data aggregation, initial settlement, final settlement, and/or financial settlement.

Work queue types are defined by a number of attributes. For each work queue type you need to create, you should determine the following attributes:

Work Queue: The work queue to which the type belongs.

Priority Level: The priority level for work queue items of this type.

Time Frame for Resolution: The amount of time by which work queue items of this type should be resolved.

Related LODESTAR Product: The Oracle Utilities product associated with the work queue type. This allows you to create work queue type specific to a particular Oracle Utilities product, such as Oracle Utilities Billing Component, Oracle Utilities Load Profiling and Settlement, or Oracle Utilities Quotations Management.

Does it Stop Billing: Should work queue items of this type stop billing processing?

Approval Only: Should work queue items of this type immediately be in “awaiting approval” mode when created, and not require resolution prior to approval?

Approval Procedure: The procedure by which work queue items of this type are approved before being closed. Each approval procedure consists of one or more ordered steps, and at each step, one or more approvals are required from users assigned a specific approval level.

Approval Time Frame: The amount of time by which work queue items of this type should be approved.

Related Business Process: The business processes associated with the work queue type. This allows you to create work queue type specific to a particular business process, such as billing, settlement, or pricing.

Does Closing the item trigger an event: Should closing work queue items of this type result in some other action or function being invoked?

Custom Parameters: In addition to the attributes listed above, you can also define custom parameters for work queue types. For example, you might want to use Account ID or Customer ID as an attribute for each work queue type. Work Queues comes installed with a number of pre-configured custom parameters, including:

- Account ID
- Process ID (an ID associated with the process that created the item)
- Rate Form
- Read Date
- Amount
- Meter ID
- Recorder ID
- Channel Number
- Interval Data Start Time
- Transaction Number (used with the Oracle Utilities Receivables Component)
- Operating Company
- Jurisdiction
- Item Data

Hints

Work queue types can also have hints associated with them. Hints provide a textual description to aid users in resolving work queue items, and can also define application links that a user can choose to follow in order to help resolve an item. For example, a link may specify a screen within the LODERSTAR Customer Choice Suite, or possibly even a URL to a third party feature or application.

You should define any hints associated with each work queue type you define. The Work Queues application includes a predefined hint (ACCTADDR) that includes a link to the Basics tab of the Account View.

Configuring Work Queues and Work Queue Types

Once you've defined the work queues and work queue types needed, the next step is to set up database records that represent the defined work queues, work queue types, and other data.

The order in which you setup the records in the database differs slightly from the order in which you define things because some records need to be in place before other records can be created. For example, work queues, Oracle Utilities products, and business processes are selected from their respective tables rather than entered as values when creating work queue types.

Setting Up Lookup Tables

Before you can create work queue types, you need to define a set of records in a number of lookup tables in the Oracle Utilities Data Repository, including:

- **LODESTAR Product Table**
- **Business Process Table**
- **Work Queue Table**
- **Approval Level Table**

Records in these tables are used as attributes for work queue type records. Create records in these lookup table using Data Manager.

LODESTAR Product Table

Records in the LODESTAR Product table define specific Oracle Utilities products which can be used in defining a work queue type. This allows you to create work queue type specific to a particular Oracle Utilities product, such as Oracle Utilities Billing Component, Oracle Utilities Load Profiling and Settlement, or Oracle Utilities Quotations Management. You should create a LODESTAR Product record for each product you intend to use in your work queues configuration. Records in the LODESTAR Product table contain the following fields:

- **Code:** A unique code used to identify the product.
- **Description:** An optional description of the product.

Business Process Table

Records in the Business Process table define business processes which can be used in defining a work queue type. This allows you to create work queue type specific to a particular business process, such as billing, settlement, or pricing. You should create a Business Process record for each process you intend to use in your work queues configuration. Records in the Business Process table contain the following fields:

- **Name:** A unique name of the business process.
- **Description:** An optional description of the business process.

Work Queue Table

Work queues can be thought of as "buckets" into which work items are grouped, and are used as an attribute of a work queue type. You should create a Work Queue record for each work queue required for your work queues configuration. Records in the Work Queue table contain the following fields:

- **Code:** A unique code for the queue.
- **Description:** An optional description of the queue.

Approval Level Table

Approval levels represent different levels of approval needed when approving work queue items. For example, you might set up approval levels for Billing Analysts, Account Managers, and System Administrators. Each of these might be assigned to one or more approval procedures. You should create a Work Queue Approval Level record for each approval level required for your work queues configuration. Records in the Work Queue Approval Level table contain the following fields:

- **Code:** A unique code for the approval level.
- **Description:** An optional description of the approval level.

Setting Up Work Queue Types

After setting up any required lookup records, the next step is to setup work queue types. Work queue types are the core entity of the Work Queues application. A work queue type record defines a type of work queue item. Setting up work queue types involves setting up three things:

- **Approval Procedures**
- **Work Queue Type Records**
- **Hints**

Use Data Manager to create the database records described below.

Approval Procedures

Each work queue type may have an approval procedure (defined in the **Approval Procedure Table**) associated with it, which defines the process by which work queue type items are approved before being closed. Each approval procedure consists of one or more ordered steps (defined in the **Approval Step Table**). At each step, one or more approvals are required from users assigned a specific approval level (defined in the **Approval Level Table**). A single approval process may be used by several work queue types.

Note: When setting up approval procedures, be sure to account for any Approval Level and corresponding values if configuring record level security . See **Chapter 7: Configuring Energy Information Platform Security** for more information about setting up record level security.

Approval Procedure Table

You should create a Work Queue Approval Procedure record for each approval procedure required for your work queues configuration. Records in the Work Queue Approval Procedure table contain the following fields:

- **Name:** A unique name for the approval procedure.
- **Description:** An optional of the approval procedure.

Approval Step Table

You should create a Work Queue Approval Step record for each approval step in each approval procedure required for your work queues configuration. Records in the Work Queue Approval Step table contain the following fields:

- **Approval Procedure:** The approval procedure to which the step belongs. Values in this field come from the **Approval Procedure Table**.
- **Step Number:** The step number for this step in the approval process.
- **Approval Level:** The approval level required to approve this step. Values in this field come from the **Approval Level Table**.
- **Number of Approvals:** The number of approvals required for this step.

Work Queue Type Records

You should create a Work Queue Type record for each work queue type required for your work queues configuration. Records in the Work Queue Type table contain the following fields:

- **Code:** A unique code for the type.
- **Description:** An optional description of the type.
- **Work Queue:** The default work queue for all work queue items of this type. Values in this field come from the **Work Queue Table**.
- **Default Priority Level:** The default priority level for all work queue items of this type. Priority levels are represented as integers, starting at “1” as the highest priority. The lower the number - the higher the priority. The default priority level of a type will be assigned to each generated item of that type unless overridden.
- **Default Work By Hours:** The default number of hours in which all work queue items of this type are to be resolved.
- **Default Product:** The default product for all work queue items of this type. This is used primarily to filter work queue items based on product. Values in this field come from the **LODESTAR Product Table**.
- **Stop Billing:** A flag that indicates if a work queue item of this type should stop billing. There are four options:
 - **Automatic Only** - Work queue items stop billing for accounts on “Fully Automatic” billing mode.
 - **Both** - Work queue items stop billing for accounts on either “Fully Automatic” or “Approval Required” mode.
 - **Manual Only** - Work queue items stop billing for accounts on “Approval Required” mode.
 - **No** - Do not stop billing (work queue item will be informational only).

This field is the same as the Stop Billing flag on the Account Note table.

- **Approval Only:** A flag that indicates that work queue items of this type require an approval only. In this case, when closed, items of this type need to indicate whether they were approved or rejected.
- **Approval Procedure:** The approval procedure (see **Approval Procedures** on page 5-12) for all work queue items of this type. If supplied, this approval process must be completed prior to an item of this type being closed, regardless of whether or not the type is “Approval Only”. Values in this field come from the **Approval Procedure Table**.
- **Approved By Hours:** The number of hours allocated for each approval, if an approval process is associated with this type. If not specified, the Work By Hours value above will be used.
- **Default Process:** The default business process for all work queue items of this type. This is used primarily to filter work queue items based on processes. Values in this field come from the **Business Process Table**.
- **Close Callback ProgId:** The program ID of a COM method to be invoked when work queue items of this type are closed. See **Close Item Callback** on page 5-15 for more information.

Custom Parameters

In addition to the core attributes listed above, you can also define custom parameters for work queue types. Doing so involves three steps:

1. **Add Columns to Work Queue Item Tables:** The first step is to add columns for each custom parameter to the Work Queue Open Item and Work Queue Closed Item tables. This is necessary to ensure that work item records include your custom parameters.

This step is performed by adding the appropriate records to the meta-data tables in the Oracle Utilities Data Repository, and should be performed by Oracle.

2. **Setup Parameters in Custom Parameters Table:** The next step is to define each custom parameter in the Work Queue Custom Parameter table. You should create a Work Queue Custom Parameter record for each custom parameter required for your work queues configuration. Records in this table contain the following fields:
 - **Column Name:** The name of the column added to the Work Queue Open Item and Work Queue Closed Item tables. This value must match the column name exactly.
 - **Filter Area Name:** The name used to define a security Area name for filtering of queries against the work queue item tables. Areas are associated to users via Positions using the Security Administration screens in the Customer Choice Suite. **Note:** This column was used by previous versions of the software, but as of v4.00 is no longer used.
 - **Show All Permission Name:** A “Show All” security permission that bypasses the Area Name Filter filter for any user that has the permission. **Note:** This column was used by previous versions of the software, but as of v4.00 is no longer used.
3. **Link Parameters to Work Queue Types:** The last step is to link the parameters in the Custom Parameters table to the appropriate work queue type records in the Work Queue Type Parameter table. You should create a Work Queue Type Parameter record for each work queue custom parameter required for each work queue type in your configuration. Records in this table contain the following fields:
 - **Work Queue Type:** The work queue type record associated with the parameter. Values in this field come from the **Work Queue Type** table.
 - **Custom Parameter:** The parameter to be linked. Values in this field come from the **Work Queue Custom Parameter** table.
 - **Required:** A flag that indicates if the parameter is required for work queue items of the type specified in the Work Queue Type field.

Custom parameters can also be configured to not be displayed on the Work Queue Search and Search Results screens through creation of a configuration file called "WQExcludeParams.XML" in the C:\LODESTAR\Web\WQ directory.

The structure of the WQExcludeParams.xml file is as follows:

```
<ExcludeParams>
  <param typecode="MAX RISK"    columnname="JURISCODE"  excludeFrom="Search"/>
  <param ypecode="MAX RISK"    columnname="OPCOCODE"  excludeFrom="Results"/>
  <param typecode="MAX RISK"    columnname="AMOUNT"    excludeFrom="Both"/>
</ExcludeParams>
```

where:

- **typecode** represents the Work Queue Type the filter applies to,
- **columnname** is the Custom Parameter Column Name that should be excluded from the page,
- **excludeFrom** represents the page from which the Custom Parameter is to be excluded ("Search", "Results", or "Both"). If the excludeFrom attribute is omitted, the specified Custom Parameter will be excluded from both the Search and Results pages.

Close Item Callback

Work Queue Types can be configured to invoke a specified COM method upon the closing of a work queue item of that type. This function is performed via a custom COM interface defined to specify how to invoke the close item callback method. This interface consists of a single method that will be invoked (once) when a work queue item is closed. The Interface Definition Language (IDL) for this interface is shown below.

```
[
    object,
    uuid("62C08908-435D-46e6-BB98-675AF6A4A9F9"),
    dual,
    helpstring("LODESTAR IWorkQueueCloseCB Interface"),
    pointer_default(unique)
]
__interface IWorkQueueCloseCB : IDispatch
{
    [helpstring("method ItemClosed")]
    HRESULT ItemClosed([in] BSTR xmlDataSource, [in] BSTR xmlWQItem);
};
```

Configuring a custom callback method consists of the following steps:

1. Create and register a COM object (DLL) that implements the work queue item close callback interface defined above, and that performs the function to be invoked upon closing of the work queue item. This DLL will need to be registered on every machine that hosts the Work Queue application code.
2. Specify the COM ProgId of the object created in step 1 in the Close Callback ProgId column for the work queue type that will invoke the method.

Example: Oracle Utilities Billing Component - Contract Management and Oracle Utilities Quotations Management can use a COM method to update the status of a contract to either APPROVED or REJECTED upon the final approval or rejection of a contract's approval. The DLL that updates the contract is called "LSCM.dll," and the method is called "WorkQueueCloseCB". To enable this update functionality, users can enter "LSCM.WorkQueueCloseCB" in the Close Callback ProgId column of any work queue type used for contract approvals.

If the Close Callback ProgId column is populated for a work queue type, when a work queue item of that type is closed for any reason, the ItemClosed method of the COM object represented by the ProgId will be invoked. The method will be invoked after the work queue item record is inserted into the Work Queue Closed Items table, passing the state of the work queue item to the ItemClosed method via the xmlWQItem parameter. The current xmlDataSource object will also be passed to the ItemClosed method in case the method needs to access the database.

The invocation of the ItemClosed method will be wrapped in the same database transaction as the action that caused the work queue item to be closed. If the ItemClosed method is unsuccessful, then the entire action (including closing the work queue item) will be rolled back.

Hints

Hints provide a textual description to aid users in resolving work queue items. Hints can also define application links that a user can choose to follow in order to help resolve an item. For example, a link may specify a screen within the LODERSTAR Customer Choice Suite, or possibly even a URL to a third party feature or application. Specific links may need to specify data values within the item that are required for the link. For example, a link to the account view screen would require the item to provide an account id.

Setting up hints involves three steps:

1. **Create Hints**
2. **Create Hint Links**
3. **Link Hints to Work Queue Types**

Each step is described below.

Create Hints

The first step is to create a record for each hint in the Work Queue Hint table. You should create a Work Queue Hint record for each work queue hint required for your work queues configuration. Records in this table contain the following fields:

- **Name:** A unique name for the hint.
- **Description:** A description of the hint.
- **Link:** An optional link associated with the hint. Links are XML strings that define the link and any necessary values or parameters associated with the link. Hint links use the XLink standard to describe the link and parameters (called 'arcs'). The namespace for providing the link itself (xl:href) and the parameter arcs (xl:from and xl:to) is xmlns:xl="http://www.w3.org/1999/xlink".

Create Hint Links

The next step (if applicable) is to create links for each hint. Hint Links contain the following, defined in an XML string:

- **Link:** The link to the page associated with the hint. Links are contained in the <link> element in the XML string. This element contains a destination (href), which can be external or internal. For example, a link to the Account Billing screen (accessible from the Account view dashboard) would look like this:

```
<link xmlns:xl="http://www.w3.org/1999/xlink" xl:href="../../../classic/viewaccount/BillView.asp" >
```

- **Parameters:** Any required parameters needed for the hint page. Parameters are contained in <param> elements in the XML string. Parameter attributes include:
 - **xl:from:** the name of the parameter in the calling page. If this field exists on the calling page, its value will be passed to the destination page.
 - **xl:to:** the name of the parameter in the destination page.
 - **type:** the type of the parameter (string, integer, float, date, datetime)
 - **default:** the default value for this parameter. If no field exists on the calling page, or the field is blank, this default value will be passed to the destination page.

For example, a parameter that passes the account id from a work queue item (ACCOUNTID) to a destination (Id) might look like this:

```
<param xl:from="ACCOUNTID" xl:to="Id"/>
```

Visit <http://www.w3.org/1999/xlink> for more information about XLink.

The following examples are hint links to several common screens in the Energy Information Platform.

Customer View (dashboard):

Displays the Customer View (splash) for a Customer ID from a Work Queue item.

```
<link xmlns:xl="http://www.w3.org/1999/xlink" xl:href="MainWebPart.aspx">
  <param xl:to="Mode" default="Customer"/>
  <param xl:from="CUSTOMERID" xl:to="Id"/>
</link>
```

Note: CUSTOMERID is not a default attribute of Work Queue items.

View Customer Accounts:

Displays the Accounts screen for a Customer ID from a Work Queue item.

```
<link xmlns:xl="http://www.w3.org/1999/xlink" xl:href="../../classic/
viewcustomer/Accounts.asp">
  <param xl:from="CUSTOMERID" xl:to="Id"/>
</link>
```

Note: CUSTOMERID is not a default attribute of Work Queue items.

Account View (dashboard):

Displays the Account View (splash) for an Account ID from a Work Queue item.

```
<link xmlns:xl="http://www.w3.org/1999/xlink" xl:href="MainWebPart.aspx">
  <param xl:to="Mode" default="Account"/>
  <param xl:from="ACCOUNTID" xl:to="Id"/>
</link>
```

Account - Recorders:

Displays the Recorders screen for an Account ID from a Work Queue item.

```
<link xmlns:xl="http://www.w3.org/1999/xlink" xl:href="../../classic/
viewaccount/Recorders.asp">
  <param xl:from="ACCOUNTID" xl:to="Id"/>
</link>
```

Account - Meters:

Displays the Meters screen for an Account ID from a Work Queue item.

```
<link xmlns:xl="http://www.w3.org/1999/xlink" xl:href="../../classic/
viewaccount/Meters.asp">
  <param xl:from="ACCOUNTID" xl:to="Id"/>
</link>
```

Account - Billing (Oracle Utilities Billing Component):

Displays the Billing screen for an Account ID from a Work Queue item.

```
<link xmlns:xl="http://www.w3.org/1999/xlink" xl:href="../../classic/
viewaccount/BillView.asp">
  <param xl:from="ACCOUNTID" xl:to="Id"/>
</link>
```

Account - Financials (Oracle Utilities Billing Component):

Displays the Financials screen for Account ID from a Work Queue item.

```
<link xmlns:xl="http://www.w3.org/1999/xlink" xl:href="../../classic/fme/
financials.asp">
  <param xl:from="ACCOUNTID" xl:to="Id"/>
</link>
```

Account - Collections (Oracle Utilities Billing Component):

Displays the Collections screen for an Account ID from a Work Queue item.

```
<link xmlns:xl="http://www.w3.org/1999/xlink" xl:href="../../classic/collections/ColAcctView.asp">
  <param xl:from="ACCOUNTID" xl:to="Id"/>
</link>
```

Usage Search:

Displays the Usage search screen.

```
<link xmlns:xl="http://www.w3.org/1999/xlink" xl:href="../../classic/idg/Search.asp">
  <param xl:from="noattr" xl:to="noparam"/>
</link>
```

View Cut Series:

Displays the Interval Data Viewer Results screen (Series tab) for a Recorder ID and Channel from a Work Queue item.

```
<link xmlns:xl="http://www.w3.org/1999/xlink" xl:href="../../classic/idg/List.asp">
  <param xl:to="O_RECORDER" default="EQ"/>
  <param xl:from="RECORDERID" xl:to="X_RECORDER"/>
  <param xl:to="O_CHANNEL" default="EQ"/>
  <param xl:from="CHANNELNUM" xl:to="X_CHANNEL"/>
  <param xl:to="Command" default="ListSeries"/>
</link>
```

View Cuts:

Displays the Interval Data Viewer Results screen (Cuts tab) for a Recorder ID and Channel from a Work Queue item.

```
<link xmlns:xl="http://www.w3.org/1999/xlink" xl:href="../../classic/idg/List.asp">
  <param xl:to="O_RECORDER" default="EQ"/>
  <param xl:from="RECORDERID" xl:to="X_RECORDER"/>
  <param xl:to="O_CHANNEL" default="EQ"/>
  <param xl:from="CHANNELNUM" xl:to="X_CHANNEL"/>
  <param xl:to="Command" default="ListCuts"/>
</link>
```

Interval Data Manager - Cut Search:

Opens the Interval Data Manager Cut Search screen (used when searching Pervasive.SQL/ *.BTE files).

```
<link xmlns:xl="http://www.w3.org/1999/xlink" xl:href="../../classic/ide/CutSearch.asp">
  <param xl:from="noattr" xl:to="noparam"/>
</link>
```

Interval Data Manager - Workset:

Opens the Interval Data Manager Workset page.

```
<link xmlns:xl="http://www.w3.org/1999/xlink" xl:href="../../classic/ide/Workset.asp">
  <param xl:from="noattr" xl:to="noparam"/>
</link>
```

Adapter - Inbound:

Opens the Adapter Inbound screen for an Adapter process from a Work Queue item.

```
<link xmlns:xl="http://www.w3.org/1999/xlink" xl:href="../../classic/adapter/
runtime.asp">
  <param xl:from="ITEMDATA" xl:to="Direct"/>
</link>
```

Run Rate Schedule:

Executes a rate schedule, passing identifier values from the Work Queue item.

```
<link xmlns:xl="http://www.w3.org/1999/xlink" xl:href="HintRate.asp">
  <param xl:to="RATE_CODE" default="<RATE_CODE>"/>
  <param xl:to="OPERATING_COMPANY" default="<OPCO>"/>
  <param xl:to="JURISDICTION" default="<JURIS>"/>
  <param xl:to="RateParams" default="<PARAM_NAME>"/>
  <param xl:to="RateParams" xl:from="<PARAM_VAL>"/>
</link>
```

where:

- **<RATE_CODE>** is the Rate Form Code of the rate schedule to execute. Only the most recent version of a rate schedule can be executed.
- **<OPCO>** is the Operating Company Code of the rate schedule to execute.
- **<JURIS>** is the Jurisdiction Code of the rate schedule to execute.
- **<PARAM_NAME>** is the name of the Rules Language identifier
- **<PARAM_VAL>** is the name of the column in the Work Queue item that contains the value to be passed to the identifier.

Link Hints to Work Queue Types

The last step is to link the hint with the appropriate Work Queue Type records in the Work Queue Type Hint table. You should create a Work Queue Type Hint record for each work queue hint required for each work queue type in your work queues configuration. Records in this table contain the following fields:

- **Work Queue Type:** The work queue type to which the hint is associated. Values in this field come from the **Work Queue Type** table.
- **Work Queue Hint:** The hint name, from the **Work Queue Hint** table.
- **Hint Order:** An integer that represents the order in which the hints appear on work items of this type. A work queue type can have more than one hint associated with it. In this case, the first hint for a work queue type uses the lowest Hint Order value.
- **Description:** A description of the Type Hint.

Creating Work Queue Items

This section describes how to set up Oracle Utilities and external applications to send messages and work queue items to the Work Queues application, including:

- **Using Oracle Utilities Billing Component**
- **Using the Oracle Utilities Rules Language**
- **Using External Applications**

Using Oracle Utilities Billing Component

Oracle Utilities Billing Component can be configured to automatically send exceptions and work queue items to the Work Queues application. Oracle Utilities Billing Component normally writes exceptions to the Account Notes table. When using the Work Queues application alongside Oracle Utilities Billing Component, exceptions can be written to the Work Queue Open Item table instead of the Account Notes table.

Configuring Oracle Utilities Billing Component to use the Work Queues application involves the following steps:

- **Setting Up Work Queue Type Records**
- **Enabling Work Queues**
- **Rules Language Configuration**

Setting Up Work Queue Type Records

This step involves creating Work Queue Type records for the types of exceptions or errors that you want to record with work queue items. Work Queue Type records that match the Account Note Type records used in previous versions of Oracle Utilities Billing Component are delivered with Oracle Utilities Billing Component and are also pre-configured in the Work Queue and Work Queue Types table, and include the following:

- **ABORT:** ABORT statement executed during bill calculation.
- **BILLED:** Account had been billed.
- **EXCEPTION:** Data missing on day after read date.
- **HOLD:** User requested hold.
- **INFO:** User entered.
- **SYSTEM REJECTED:** Failure during billing - validation or calculation.
- **USER REJECTED:** User rejected during manual review.
- **USERCALC:** User computed bill via Current Bill.
- **WARNING:** Warning statement executed during bill calculation.

If your Oracle Utilities Billing Component implementation requires additional work queue types, you must create the appropriate Work Queue Type records for them. Work queue items based on these additional (non-Account Note) work queue types must be created via the Oracle Utilities Rules Language configuration used by Oracle Utilities Billing Component.

When creating Work Queue Types for use with Oracle Utilities Billing Component, the **Stop Billing** flag should be set appropriately, based on your desired behavior. This flag provides four options:

- **Automatic Only** - Work queue items stop billing for accounts on “Fully Automatic” billing mode.

- **Both** - Work queue items stop billing for accounts on either “Fully Automatic” or “Approval Required” mode.
- **Manual Only** - Work queue items stop billing for accounts on “Approval Required” mode.
- **No** - Do not stop billing (work queue items will be informational only).

See **Setting Up Work Queue Types** on page 5-12 for more information about creating Work Queue Type records.

Enabling Work Queues

To enable the Work Queues application to work with Oracle Utilities Billing Component, include the **USE_WORK_Q_FOR_ACCOUNT_NOTES = 1** parameter in the **LODESTAR.CFG** file. See **LODESTAR.CFG** on page 2-2 for more information.

Rules Language Configuration

Once the Work Queues functionality has been enabled, Oracle Utilities Billing Component will create work queue items based on Account Notes automatically. Work queue items based on other (non-Account Note) work queue types must be created via the Oracle Utilities Rules Language configuration used by Oracle Utilities Billing Component.

When creating Work Queue items for use with Oracle Utilities Billing Component, the following are required in order for the work queue items to prevent billing (based on the value of the **Stop Billing** flag):

- The **Account ID** column must be populated with the account for which the exception occurred
- The **Read Date** column must be populated with the read date for the billing period being processed.

See **Using the Oracle Utilities Rules Language** on page 5-22 for more information about configuring the Oracle Utilities Rules Language to create work queue items.

Using the Oracle Utilities Rules Language

You can also use the Oracle Utilities Rules Language to open work queue items, via either the WQ_OPEN function or the SAVE TO TABLE statement.

Both can be triggered in the Rules Language through use of an IF THEN ELSE statement. In this case, if a certain condition or situation arises or exists, the Rules Language opens the item.

Both approaches use an identifier as a single argument. The identifier is a stem that should contain several tail attributes, as described below. Identifiers in parentheses must be used when using the SAVE TO TABLE statement.

Identifier	Description
TYPE (WQTYPECODE)	The work queue item type. The type defines a number of attributes for the work queue item. Required.
QUEUE (WQQUEUECODE)	The work queue. If not supplied, the default queue for the TYPE is used.
PRODUCT (PRODUCTCODE)	<i>Optional.</i> The Oracle Utilities product associated with the work queue item. If supplied, the PRODUCT must have a corresponding record in the LODESTAR Product table. If not supplied, the default product for the TYPE is used.
PRIORITYLEVEL	<i>Optional.</i> The priority level for the work queue item. If not supplied, the default PRIORITYLEVEL for the TYPE is used.
WORKBYTIME	<i>Optional.</i> The time by which the item is expected to be resolved. If not supplied, it is calculated from the Default Work By Hours values of the TYPE.
ASSIGNEDTOUSERID	<i>Optional.</i> The UserId of the user to which the item is assigned. This value will change each time the item is unassigned, reassigned, resolved, approved, rejected, or closed. If supplied, the User ID must have a corresponding record in the Users table in the Security database.
PROCESSNAME	<i>Optional.</i> The business process associated with the work queue item. If supplied, the PRODUCT must have a corresponding record in the Business Process table. If not supplied, the default process for the TYPE is used.
ACCOUNTID	<i>Optional.</i> The Account ID associated with the work queue item (if applicable). Note: When creating work queue items for use with Oracle Utilities Billing Component (to prevent billing for an account until the work queue item is closed), the Account ID is required .

Identifier	Description
READDATE	<p><i>Optional.</i> The Read Date associated with the work queue item (if applicable). This attribute is used primarily when creating work queue items for use with Oracle Utilities Billing Component.</p> <p>Note: To prevent billing for an account during a specific bill period until the work queue item is closed, the Read Date is required.</p>

In addition to these, the column name of any custom parameters used by the work queue type may also be specified as a tail identifier. See the **Custom Parameters** on page 5-14 for more information about custom parameters.

Using the WQ_OPEN Function

In addition to opening a work queue item record in the Work Queue Open Item table, the WQ_OPEN function returns an XML document containing the opened work queue item.

Example:

Open an ERROR type work queue item in the "QUEUE_1" work queue if a cut of interval data contains more than 50 missing intervals.

```
IF MISSING_INT > 50
THEN
    ERROR.TYPE = "ERROR";
    ERROR.QUEUE = "QUEUE_1";
    ERROR.PRODUCT = "LPSS";
    ERROR.ASSIGNEDTOUSERID = "lou_p";
    OPEN_ERROR_WQ = WQ_OPEN(ERROR);
END IF;
```

See the description of the **WQ_OPEN Function** on page 13-20 in the Oracle Utilities Rules Language Reference Guide for more information about using this function.

Using the SAVE TO TABLE Statement

Using the SAVE TO TABLE statement opens a work queue item record in the Work Queue Open Item table, but does not return the opened work queue item. In addition, tail identifiers must exactly match the column names from the Work Queue Open Item table.

Example:

Open an ERROR type work queue item in the "QUEUE_1" work queue if a cut of interval data contains more than 50 missing intervals.

```
IF MISSING_INT > 50
THEN
    ERROR.WQTYPECODE = "ERROR";
    ERROR.WQQUEUECODE = "QUEUE_1";
    ERROR.PRODUCTCODE = "BX";
    ERROR.ASSIGNEDTOUSERID = "lou_p";
    SAVE_ERROR_TO_TABLE_LSWQOPENITEM;
END IF;
```

See the description of the **Save Statements** on page 6-3 in the *Oracle Utilities Rules Language Reference Guide* for more information about using this statement.

Using External Applications

Work queue items can also be opened from an external application using the Open method of the Work Queues interface.

Chapter 6

Setting Up, Configuring, and Running the Energy Information Platform Adapter

The Energy Information Platform Adapter (referred to as "Adapter" for short) is a component of the Oracle Utilities Energy Information Platform that enables validation, import, and export of data to/from the Oracle Utilities Data Repository.

This chapter describes the process of setting up, configuring, and running the Adapter, including:

- **Defining the Adapter Implementation**
- **Installing the Adapter Software**
- **Configuring the Adapter**
- **Using Adapter Data Converters**
- **Implementing Adapter Services and Business Rules**
- **The Adapter Server**
- **The Adapter Monitor**
- **Securing and Encrypting the Adapter Server and Monitor**

Defining the Adapter Implementation

The first step in setting up and configuring the Adapter is to define the Adapter implementation you wish to configure. This includes:

- **Defining Business Requirements**
- **Defining and Naming Adapter Servers**

Defining Business Requirements

Defining business requirements is the process of identifying the specific business functions you wish to perform with the Adapter. For each business requirement you plan to implement, you must also define the following:

- **Business Rule Types**
- **Data Formats**
- **Runtime Service Types**
- **Business Rule Processing**

Business Rule Types

The Adapter can import and/or validate data (relational or interval), export data, execute rate schedules configured in the Oracle Utilities Rules Language, and call custom COM objects. There are six types of business rules that can be implemented using the LODESETAR Adapter. Each is described below.

Import

Import business rules are used to import relational data in to the Oracle Utilities Data Repository. Import business rules can perform validations and other functions on the data during import. Import business rules can also be used to validate relational data without importing the data.

Import/Interval

Import/Interval business rules are used to import relational and interval data in to the Oracle Utilities Data Repository. Import/Interval business rules can perform validations and other functions on the data during import. Import/Interval business rules can also be used to validate relational and/or interval data without importing the data.

Interval

Interval business rules are used to import interval data in to the Oracle Utilities Data Repository that is in either the Oracle Utilities Enhanced Input/Output (*LSE) format or an XML format that is compatible with the Oracle Utilities Interval Data interfaces (LSINTD). Interval business rules allow import of interval data to either standard or enhanced interval data tables. Interval business rules can perform validations and other functions on the data during import.

Assembly

Assembly business rules are used to call custom .NET assemblies through the Adapter. See **Implementing Assembly Services and Business Rules** on page 6-51 for more information about implementing Assembly business rules.

COM

COM business rules are used to call custom COM objects through the Adapter. If you have a particular interface that has been used and tested before, and you would like to use it in the Adapter, then you can simply create a wrapper interface that works with the Adapter. Interfacing with the Adapter requires that the user create a COM method that follows specified standards for interfacing with the Adapter COM type. See **Implementing COM Services and Business Rules** on page 6-48 for more information about implementing COM business rules.

CustomQuery

CustomQuery business rules are used to call custom stored procedures through the Adapter. See **Implementing CustomQuery Services and Business Rules** on page 6-57 for more information about implementing CustomQuery business rules.

Rate

Rate business rules are used to execute rate schedules configured in the Oracle Utilities Rules Language from the Adapter. See **Implementing Rate Services and Business Rules** on page 6-59 for more information about implementing Rate business rules.

Meter Data Management Business Rules

Meter Data Management business rules are used with Oracle Utilities Meter Data Management to import usage readings and perform other tasks. See **Oracle Utilities Meter Data Management Business Rule Types** on page 5-11, **Configuring Usage Data Import using the Adapter** on page 5-36, and **Oracle Utilities Meter Data Management Web Services** on page 7-2 in the *Oracle Utilities Meter Data Management Installation and Configuration Guide* for more information about implementing Meter Data Management business rules.

Data Formats

You also need to identify or define the format(s) of the data that will be processed by the Adapter. The Adapter can process data in any delimited or XML format, and is not limited to Oracle Utilities standard formats.

The primary consideration to address when defining data formats for use with the Adapter is the source of the data. The source of the data to be processed can be a key factor when implementing the Adapter, since the source will determine how the data is delivered to the Adapter, and the format of the data itself. For example, if your incoming data is produced by a system that posts the data as files in a directory on a file system and produces data in a comma-separated delimited format, your Adapter implementation will have to be configured to:

- a. Pick up the data from the appropriate directory in the file system, and
- b. Convert the data from its delimited format into an XML format that the Adapter can process.

Note: When processing payload data in XML format, the Adapter supports standard XML CDATA sections, as well as encoded entity references, such as “'” for a single straight quote (“ ’”).

Delivery Methods

The Adapter can accept payload from a file system, a table in the Oracle Utilities Data Repository, or a Java Messaging System (JMS) queue. The delivery method used by a business rule is determined by its Runtime Service Type. See **Runtime Service Types** below for more information.

Data Conversion

As noted above, the Adapter can process data in any delimited or XML format. When processing data in a delimited format, the Adapter must first convert the data into an XML format. This can be performed through use of data converters supplied with the Adapter, which include:

- Basic CSV parsing
- Column Counting CSV Parsing
- Row Identifying CSV Parsing
- Row and Column Naming CSV Parsing
- Parsing using XSLT Maps

See **Using Adapter Data Converters** on page 6-22 for more information about using data converters with the Adapter.

Runtime Service Types

Business rules are run by services running on one or more Adapter servers (see **Defining and Naming Adapter Servers** on page 6-6 for more information). The runtime service type defines the manner in which the payload to be processed by the Adapter is delivered. The Adapter can run three types of services. Each is described below.

Database

Database services pick up payload from the Ice Payload or Payload Extension table in the Oracle Utilities Data Repository. Database services require a custom payload type be defined in the Payload Type table. See **Payload Types** on page 11-7 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about payload types. The Runtime Service Type for database services is **DPORTAL**.

File

File services pick up payload from a specified directory on the Adapter server or network. The Runtime Service Type for file services is **FPORTAL**.

Note: Runtime services of type FPORTAL (file) that monitor a network location cannot be run on more than one application server at the same time.

Queue

Queue services pick up payload from a Java Messaging Service (JMS) queue set up on a JMS server. Queue services require JMS server(s) and queue(s) be installed and configured. See **Installing the Adapter Software** on page 6-7 for more information about installing JMS servers to work with the Adapter. The Runtime Service Type for queue services is **QPORTAL**.

MDM Web Transactions

MDM web transaction services execute operations performed by Oracle Utilities Meter Data Management users working with the application's web-based user interface, including uploading usage via LSM files, editing and saving changes, and manually validating approving usage readings. The Runtime Service Type for MDM web transactions is **MDMPORTAL**.

Note: At least one service of type MDMPORTAL must be running in order for users to use the Oracle Utilities Meter Data Management application via the web-based user interface.

Note: MDMPORTAL services are used **ONLY** by Oracle Utilities Meter Data Management.

Business Rule Processing

Defining business rule processing involves defining what specific actions or functions each of the business requirements will perform. Business rule processing can include validations, operations, and global functions.

Validations

Validations allow to check that the incoming data matches specified criteria. Validations are also referred to as "Table Validations", as they are defined when incoming data is "mapped" to its destination table in the Oracle Utilities Data Repository. You can perform validations this as part of the import process, or as a separate operation. For business rules that will perform validations, you should define those validations in general terms. For example:

- If you are importing account data, you might want to validate that the customer associated with the imported accounts already exist in the Oracle Utilities Data Repository.

- If you are importing interval data, you might want to validate that the Unit-of-Measure (UOM) and Seconds-per-Interval (SPI) values of the incoming interval data matches what is expected.

The specifics of how the Adapter will perform the validation is done during configuration (see **Configuring the Adapter** on page 6-9). See **Define Table Validations** on page 11-81 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about operations or Table Mappings.

Operations

Operations are specific functions that can be applied to the incoming data during the import process. Operations are also referred to as “Column Mappings”, as they are defined when incoming data is “mapped” to its destination column(s) in the Oracle Utilities Data Repository. For example, if you use specific naming conventions for Account or Meter IDs in the Oracle Utilities Data Repository, you can configure operations that automatically adjust Account or Meter IDs to match that convention. This can be as simple as appending a suffix to the end of an ID (such as MTR for meters) or as complex as concatenating a string of values together to form an ID. See **Define Column Mappings** on page 11-91 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about operations or Table Mappings.

Again, at this point you should define operations in general terms. You don't need to define the specifics of an operation until configuring the business rules that will use it. See **Configuring the Adapter** on page 6-9 for more information.

Global Functions

As you define the validations and operations you plan to perform with the Adapter, make a note of any that are used in multiple business rules. These operations and validations can be defined as “Global Functions”. Global functions are snippets of application code that can be used with multiple business rules. Each global function is related to a particular Business Rule Type (either Import and Import Interval, see **Business Rule Types** below), and is available during configuration of business rules of the same type.

Again, at this point you should define global functions in general terms. You don't need to define the specifics of the application code that executes the function until configuring the business rules that will use the global functions. See **Configuring the Adapter** on page 6-9 for more information.

A Note about Business Rules and Incoming Data Formats

Each business rule implemented in the Adapter is configured to use incoming data of a specific data format. In other words, if you need to import data from multiple sources in multiple data formats, you would need to implement individual business rules for each data format. For example, if you plan to import account data in both XML and comma-separated formats, you would have to configure two business rules, one for each format. This is where implementing specific validations or operations as global functions can save time and effort when implementing the Adapter.

Example: Importing Meter Read Data

The following example illustrates how you might define a simple business requirement that might be part of a larger implementation.

Description of Business Requirement: Import scalar meter read data into the Oracle Utilities Data Repository. The data to be imported comes from a meter data processing system which produces the data in XML format files. During the import process, validate that the meters associated with the meter read data exist in the Oracle Utilities Data Repository prior to inserting the data.

Let's examine the above requirement and define it in Adapter terms:

- Since we're importing scalar meter read data (relational data), the Business Rule Type would be **Import**.
- Because the data is coming into the Adapter in XML format we don't need a data converter.
- Because the data is coming into the Adapter as files posted to a file system, the Runtime Service Type would be **File** (FPORTAL).
- To handle the validation of meter reads against meters in the Oracle Utilities Data Repository, we need to configure a validation.

Based on the above, the following paragraph repeats the above description, but include parenthetical notes about how this business requirement would be defined in Adapter terms:

Description of Business Requirement: Import meter read data into the Oracle Utilities Data Repository (**Business Rule Type:** Import). The data to be imported comes from a meter data processing system which produces the data in XML format files (**Data Format:** XML files, **Runtime Service Type:** File (FPORTAL)). During the import process, validate that the meters associated with the meter read data exist in the Oracle Utilities Data Repository prior to inserting the data (**Business Rule Processing:** Validate meter ID against records in the Meter table prior to insert).

Defining and Naming Adapter Servers

After defining the business requirements as described above, you should define the specific application servers that will run the business rules and services that will address each requirement.

Adapter Servers

Adapter servers are application servers that run the Runtime Services that process the Adapter business rules. The volume of data to be processed and the number of services that need to run are the key factors in determining the number of servers you need to set and configure.

JMS Servers and Queues

JMS servers are Java Messaging Service -compliant servers that host JMS queues. These are the queues used by "Queue" (QPORTAL) type Runtime Services. In other words, if you identified the Runtime Service Type as "Queue (QPORTAL)" for any of your business requirements, you will be using JMS servers and queues.

If your Adapter implementation will make use of Java Messaging Service (JMS) servers and queues, you need to define the JMS servers and queues that will be used in your implementation. Similar to the case with Adapter servers, the volume of data to be processed and the number of queues needed to process the data are the key factors in determining the number of JMS servers/queues you need to set and configure.

Installing the Adapter Software

Installing the Adapter involves the following steps:

- **Install the Adapter Software**
- **Set Up the LTMH.CFG.XML File**
- **Install JMS Servers and Queues (Optional)**

Install the Adapter Software

Install the Adapter software as described in **Chapter 4: Installing the Oracle Utilities Application Software** in the *Oracle Utilities Energy Information Platform Installation Guide*. The Adapter is a component of the Energy Information Platform, and is installed by default when installing other Oracle Utilities products.

Install the Adapter software on each application server you plan to use as an Adapter server. Be sure to install **Application Server** components (at a minimum) when installing the software.

Set Up the LTMH.CFG.XML File

The LTMH.CFG.XML file is a configuration file used to specify the database connection used by the Adapter. The LTMH.CFG.XML file must be installed in the **C:\LODESTAR\CFG** directory on the application server on which the Adapter will run.

Set up this file on each application server you plan to use as an Adapter server.

LTMH.CFG.XML Example

```
<LTMH JMSENABLED="false">
  <!-- Uncomment and customize lines appropriate to your installation -->
  <!-- Connection for an Oracle database -->
  <JDBC URL="jdbc:oracle:thin:@localhost:1521:orcl">
    <DRIVER TYPE="ORACLE">oracle.jdbc.OracleDriver</DRIVER>
    <QUALIFIER>tr450</QUALIFIER>
    <USERNAME>tr450</USERNAME>
    <PASSWORD>password</PASSWORD>
  </JDBC>
  <!-- Optional database connection information
    - Service property settings will override these
    - Uncomment the appropriate section(s) -->
  <DEFAULT>
    <ORACLECONNECTION>
      <DSN>local</DSN>
      <QUALIFIER>tr450</QUALIFIER>
      <USERNAME>tr450</USERNAME>
      <PASSWORD>password</PASSWORD>
    </ORACLECONNECTION>
  </DEFAULT>
</LTMH>
```

See **LTMH.CFG.XML** on page 2-46 for more information about setting up this file.

Install JMS Servers and Queues (Optional)

If your implementation of the Adapter will use JMS queues, you must also install and configure JMS servers and queues. This involves the following steps:

- **Install JMS Server**
- **Configure JMS Queues**

Install JMS Server

Install the JMS server software on each application server you plan to use as JMS servers in your Adapter implementation. Refer to the JMS server documentation for specific details regarding installation and configuration.

In addition, you must create a record in the JMS Server table for the server. See **JMS Servers** on page 11-4 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about setting up JMS Server records.

Once the JMS server software has been installed and JMS Server record created, perform the following steps **for each Adapter server**:

1. Copy the JMS server's jar file onto each application server that will run the Adapter. You can copy the jar file to the C:\LODESTAR\LTMH\RUNTIME\lib directory, or any other directory on the server.
2. Create Server and System Properties records for each application server. The System Properties record should indicate the JMS server to be associated to the application server. As part of this step, be sure to edit the Class Path field on the System Properties record to include the path to the JMS server's jar file. See **Servers** on page 11-2 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about setting up server records. See **System Properties** on page 11-12 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about setting up system properties records.
3. If applicable, update the .bindings file in the C:\LODESTAR\LTMH\RUNTIME\jndi directory. The format of the bindings file varies by JMS server.

Note: Web Logic JMS servers do NOT use a .bindings file.

Note: The above steps **MUST** be performed for **EACH** Adapter server (i.e. each application server that will run the Adapter) in order for the Adapter to properly interface with the JMS servers and queues.

Configure JMS Queues

After the JMS server software has been installed on the application servers, configure the specific JMS queues that the Adapter will interface with. Refer to your JMS server documentation for details regarding configuration of JMS queues.

Also, you must create a record in the JMS Queues table for each JMS queue you will be using in your implementation. See **JMS Queues** on page 11-5 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about setting up JMS queue records.

Configuring the Adapter

After defining your Adapter implementation and installing the Adapter software (including JMS server software if applicable), the next step is to configure the Adapter to run as you defined it, meaning that the business rules you defined will run on the servers you defined. During this step you will take the business requirements you defined in general terms and translate them into specific business rules that can be run by the Adapter.

This step involves creating records in the Oracle Utilities Data Repository using the Oracle Utilities Energy Information Platform, and includes

- **Configuring Adapter Components**
- **Configuring Business Rules**
- **Configuring Adapter Services**

Configuring Adapter Components

The Adapter utilizes several components such as servers, Java Messaging Service (JMS) servers (optionally), and other components. Adapter components include:

- **Servers**
- **System Properties**
- **JMS Servers** (if applicable)
- **JMS Queues** (if applicable)
- **Payload Types** (if applicable)
- **Origination Systems**

In most cases, these components are shared by most or all of the business rules run by the Adapter. Configuring these components establishes the foundation upon which the Adapter services and business rules will run.

Servers

Servers are the application servers that run the individual services and business rules of your Adapter implementation.

Create a record in the Server table for each server you defined for your implementation (see **Defining and Naming Adapter Servers** on page 6-6). Records in the Server table contain the following fields:

- **Name:** The network name of the server.

See **Servers** on page 11-2 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about setting up server records.

System Properties

System Properties represent specific Adapter servers at runtime. System properties link **Servers** with **JMS Servers** and provide basic information which the server needs to run properly.

Create a record in the System Properties table for each server you defined for your implementation (see **Defining and Naming Adapter Servers** on page 6-6). Records in the System Properties table contain the following fields:

- **JMS Server:** The JMS server that the Adapter will use to route messages. If a JMS server is not in use, a “dummy” entry is required here.
- **Server:** The server to which the system properties apply.
- **JDK Path:** The file path to the Java executable.

- **Maximum Retry:** The maximum number of times the server should restart after an error (database disconnection, JMS failure, etc.) before terminating.
- **Debug Level:** The level of debugging information produced by the Adapter. See **Oracle Utilities Energy Information Platform Adapter Logging** on page 14-8 for more details concerning Adapter log files.
- **Map Cache Size:** The number of elements cached by the LTMH XML Map Service. Caching stylesheets allows for improved performance. By default, the system caches five (5) stylesheets. **Note:** This parameter is not used by the Adapter.
- **Retry Interval:** The number of seconds to wait after encountering a serious exception.
- **JVM Parameter:** Specific parameters for the Java Virtual Machines (JVM) used by the server.
- **Class Path:** A set of paths to directories (or to archive files) that control the classes that are available within the JVM.

See **System Properties** on page 11-12 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about setting up system properties records.

JMS Servers

JMS Servers are Java Messaging Service (JMS) compliant components used by the Adapter (optionally). Note that even if JMS Servers are not being used, a placeholder record is required.

If your implementation will use JMS servers and queue, create a record in the JMS Servers table for each JMS server you defined for your implementation (see **JMS Servers and Queues** on page 6-6). Records in the JMS Servers table contain the following fields:

- **Name:** The name of the JMS server, from the selection key in the .bindings file in the jndi directory. See the **Install JMS Server** on page 6-8 for more information about this file.

See **JMS Servers** on page 11-4 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about setting up JMS server records.

JMS Queues

JMS Queues are message queues used by JMS compliant components used by the Adapter (optionally).

If your implementation will use JMS servers and queue, create a record in the JMS Queues table for each JMS queue you defined for your implementation (see **JMS Servers and Queues** on page 6-6). Records in the JMS Queues table contain the following fields:

- **Name:** The name of the JMS queue. If certain JMS queues have been registered with a secured JMS implementation, the Versions must match those registered.
- **Description:** A description of the JMS queue.

See **JMS Queues** on page 11-5 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about setting up JMS queue records.

Payload Types

Payload types represent different payloads used by Adapter transactions.

If your implementation will use “Database” services (see **Runtime Service Types** on page 6-4), create a record in the Payload Types table for each payload type you defined for your implementation. Records in the Payload Types table contain the following fields:

- **Payload Type:** The name of the payload type.
- **Payload Type Description:** A description of the payload type.

See **Payload Types** on page 11-7 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about setting up payload type records.

Origination Systems

Origination Systems define the external data sources used by the Adapter. Though they are used primarily for reporting purposes, you must create at least one Origination System record. Records in the Origination Systems table contain the following fields:

- **System Name:** The name of the originating system.

See **Origination Systems** on page 11-10 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about setting up origination system records.

Configuring Business Rules

After configuring the Adapter components required for your implementation, the next step is to configure the business rules that will address the business requirements you defined previously (see **Defining Business Requirements** on page 6-2). Configuring business rules is the means by which you define specifically how the Adapter will perform the functions needed to meet your business requirements, and involves the following:

- **Create and Configure Global Functions**
- **Create and Configure Rule Description Language**
- **Create and Configure Business Rule Records**
- **Associate Business Rules to Rule Description Language**

Whereas Adapter components (such as servers, JMS servers, etc.) are shared by most or all of the business rules run by the Adapter, most business rules are unique in that each performs a distinct function. For this reason, much of your business rules configuration will apply to only a single business requirement.

Create and Configure Global Functions

Global functions are validations and/or operations that can be used in multiple business rules. Global functions are used **Import** and **Import/Interval** business rules (see **Business Rule Types** on page 6-2 for more information). Global Functions are not used with **Interval**, **COM**, or **Rate** business rules.

If your implementation will make use of global functions, create a record in the Global Function table for each global function you defined (see **Global Functions** on page 6-5) for your implementation. Records in the Global Functions table contain the following fields:

- **Function Name:** The name of the global function.
- **Description:** A description of the global function.
- **Business Rule Type:** The type of business rule to which the global function is related. This can be either **Import** or **Import Interval**.

See **Global Functions** on page 11-55 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about creating global function records.

Global Function Versions

For each global function you create, you must also create a record in the Global Function Version table that represents the specific version of the global function that the Adapter will use during runtime. Global function versions contain the actual function code that is executed during runtime. Records in the Global Function Versions table contain the following fields:

- **Function Name:** The name of the global function, from the parent Global Function record.
- **Start Date:** The date and time at which the global function version goes into effect.
- **Stop Date:** The date and time after which the global function version is no longer in effect.

- **Function Type:** The language the function is written in. This can be either VB script or Java script.
- **Note:** A description of what the function will do.
- **Syntax:** The calling syntax of the function. The Function Name **must** be the same as the function being called.
- **Code:** The function code that is executed at runtime. The Function Name **must** be the same as the function being called.

See **Adding Global Functions** on page 11-57 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about creating global function versions.

Create and Configure Rule Description Language

Rule Description Language is a script type that allows users to define data validations and locations in the Oracle Utilities Data Repository by mapping data from an inbound transaction to specific database tables.

Rule Description Language scripts define specific units of work for business rules, and describe what actions will be taken with an incoming payload, such as whether or not to insert particular pieces of data into the Oracle Utilities Data Repository based on a set of validations.

Rule Description Language (RDL) is required for **Import** and **Import/Interval** business rules, and is optional for **Interval** business rules (see **Business Rule Types** on page 6-2 for more information). RDLs are not used with **COM** or **Rate** business rules.

Rule Description Language scripts require Rule Description Language records, RDL Version records, and RDL Configurations.

Rule Description Language Records

Create a record in the Rule Description Language table for each **Import**, **Import/Interval**, and (optionally) **Interval** business rule you defined (see **Business Rule Types** on page 6-2) for your implementation. Records in the Rule Description Language table contain the following fields:

- **RDL Name:** The name of the RDL.
- **Rule Type Code:** The type of business rule to which the RDL is related. This can be either Import, Import/Interval, Import/Usage (used with Oracle Utilities Meter Data Management) or Interval (see **Business Rule Types** on page 6-2 for more information about business rule types).

See **Rule Description Language** on page 11-58 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about creating Rule Description Language records.

RDL Versions

RDL Versions represent specific versions of an RDL. RDL versions contain the actual script that is used during runtime. Note that only one RDL version can be active at one time.

Create a record in the RDL Version table for each RDL you created for your implementation. Records in the RDL Version table contain the following fields:

- **RDL Name:** The name of the RDL, from the parent RDL record.
- **Start Time:** The date and time at which the RDL version goes into effect.
- **Stop Time:** The date and time after which the RDL version is no longer in effect.
- **Note:** A note regarding the RDL version.

See **Adding RDL Versions** on page 11-62 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about creating RDL Versions.

RDL Configuration

As noted above, RDL Versions contain the actual RDL scripts used during runtime that perform the actual processing of the business rules you plan to run using the Adapter. RDL configuration is the process of defining those functions and operations using RDL.

For each RDL version, configuring the RDL involves the following steps:

- **Define Template Source and Destination:** This step defines the structure of the data being processed by the business rule.
 - The **Template Source** is a representation (in XML format) of what the incoming XML payload will look like when it is processed by the Adapter.
 - The **Template Destination** is a representation of the tables and columns in the Oracle Utilities Data Repository that the incoming data will be mapped to.

See **Define Template Source and Destination** on page 11-70 in the *Oracle Utilities Energy Information Platform User's Guide* for more information.

This step does not apply to RDLs with a Rule Type Code of "Interval."

- **Define RDL Mappings:** RDL mappings indicate how data defined in the Template Source map to tables in the columns in the Oracle Utilities Data Repository defined in the Template Destination. There are two types of RDL mappings:
 - **Table Mappings:** indicate how data elements map to specific tables in the Oracle Utilities Data Repository.
 - **Column Mappings:** indicate how data elements or attributes map to specific columns in the Oracle Utilities Data Repository.

See **Define RDL Mappings** on page 11-75 in the *Oracle Utilities Energy Information Platform User's Guide* for more information.

This step does not apply to RDLs with a Rule Type Code of "Interval."

- **Define Table Actions:** Table actions specify the action to be performed on the table when the specified source element is reached during processing. Action types include:
 - (Empty) – No action is performed
 - **Insert:** Inserts the record into the database
 - **Update:** Updates the record in the database
 - **Insert or Update:** Inserts the record if it does not exist, otherwise it updates the record
 - **Insert Cut:** (Import Interval only) Inserts the cut of interval reads
 - **Insert Read:** (Import Interval only) Stores off the interval read until the Insert Cut is processed
 - **Debug Output:** Sends to the output log all values that have been currently been mapped to the Template Destination.

See **Define Table Actions** on page 11-79 in the *Oracle Utilities Energy Information Platform User's Guide* for more information.

This step does not apply to RDLs with a Rule Type Code of "Interval."

- **Define Table Validations:** Table validations are validations performed before the table action. Validation types include:
 - **Stored Procedure:** a database stored procedure that returns a "TRUE" or "FALSE" string.
 - **SQL:** A SQL statement that returns TRUE if records are returned from SQL statement, and returns FALSE if no records are returned

- **SQL (No Records):** A SQL statement that returns FALSE if records are returned from SQL statement, and returns TRUE if no records are returned
- **JavaScript:** A JavaScript that returns Boolean or string “True” or “False”.
- **VB Script:** A Visual Basic script that returns Boolean or string “True” or “False”.
- **JavaScript Evaluation:** A JavaScript that returns Boolean or string “True” or “False”.
- **VB Script Evaluation:** A Visual Basic script that returns Boolean or string “True” or “False”.
- **Global Function:** JavaScript or VB Script global function that returns Boolean or string “True” or “False”.
- **Rate Schedule:** A Oracle Utilities Rules Language rate schedule that returns “PASS” (True) or “FAIL” (False).

See **Define Table Validations** on page 11-81 and **Validation Types and Parameters** on page 11-82 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about validation types.

This step does not apply to RDLs with a Rule Type Code of “Interval.”

- **Define Column Mappings:** Column mappings indicate how data elements or attributes map to specific columns in the Oracle Utilities Data Repository. Mapping types include:
 - **Direct:** the value from the Template Source is set to the selected column of the Template Destination.
 - **Stored Procedure:** the value from the RETURN parameter from the stored procedure call is set to the selected column of the Template Destination.
 - **SQL:** the return value from SQL statement is set to the selected column of the Template Destination.
 - **JavaScript:** the return value from function call is set to the selected column of the Template Destination.
 - **VB Script:** the return value from function call is set to the selected column of the Template Destination.
 - **JavaScript Evaluation:** the return value from function call is set to the selected column of the Template Destination.
 - **VB Script Evaluation:** the return value from function call is set to the selected column of the Template Destination.
 - **Global Function:** the return value from function call is set to the selected column of the Template Destination.
 - **Rate Schedule:** the return value(s) from a Oracle Utilities Rules Language rate schedule is set to specified columns in the Template Destination.

See **Define Column Mappings** on page 11-91 and **Column Mapping Types and Parameters** on page 11-93 in the *Oracle Utilities Energy Information Platform User's Guide* for more information.

This step does not apply to RDLs with a Rule Type Code of “Interval.”

- **Validate RDL:** Validating the RDL performs a validation on all SQL, Stored Procedures, and scripts in mappings and validations and returns any errors or warnings that are found.

See **Validate RDL** on page 11-101 in the *Oracle Utilities Energy Information Platform User's Guide* for more information.

- **Define Interval Data Validations:** Interval data validations are validations performed when processing interval data. Validation types include:

- **Validate Intd:** validates start/stop time and number intervals and missing intervals
- **Validate SPI:** validates SPI in processed cuts
- **Validate UOM:** validates UOM in processed cuts
- **Time Zone Conversion:** converts time zone of cut
- **Behaviors:** specifies interval data behavior, similar to options defined in INTDCONFIG.CFG.XML configuration file

See **Define Interval Data Validations** on page 11-102 in the *Oracle Utilities Energy Information Platform User's Guide* for more information.

This step applies only to RDLs with a Rule Type Code of Import Interval and "Interval."

Create and Configure Business Rule Records

Business rules are individual business rules that are available to the Adapter. In specific, they define the validations and post/retrieval logic that will be applied to data being imported/exported into/out of the Oracle Utilities Data Repository.

Create a record in the Business Rule table for each business requirement you defined (see **Defining Business Requirements** on page 6-2) for your implementation. Records in the Business Rule table contain the following fields:

- **Class Name:** Class name (package) to class for JAVA type business rules.
- **Rule Name:** The name of the business rule.
- **Description:** A description of the business rule.
- **Rule Type:** The type of business rule (from the Business Rule Type table). The Rule Type can be Assembly, COM, CustomQuery, Import, ImportInterval, Interval, Rate, or one of the Oracle Utilities Meter Data Management business rules (used with Oracle Utilities Meter Data Management only). See **Business Rule Types** on page 6-2 for more information about business rule types.
- **Optional Parameters:** Values that can be passed to Rate, Assembly, COM, or CustomQuery business rules.
- **Properties:** Properties associated with the business rule. Properties are defined in an XML structure.

See **Business Rules** on page 11-63 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about creating and configuring business rule records.

Business Rule Properties and Parameters

Business Rule Properties and Parameters define how business rules function during runtime. They define things such as whether or not the Adapter engine keeps the XML data that is to be sent to the database engine, the mode in which database operations are performed, and other specifics of how the business rule should run.

Each type of business rule use different properties and/or parameters. See **Defining Properties and Parameters for Adapter Business Rules** on page 11-114 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about business rule properties.

Implementing Type-Specific Business Rules

Though most business rules share many characteristics, the specifics of how to implement a given business rule are largely determined by the Business Rule Type. In other words, the specific steps involved in implementing business rule of one type (such as Import) can differ significantly from implementing a business rule of another type (such as COM or Rate). See **Implementing Adapter Services and Business Rules** on page 6-43 for more information about implementing business rules of specific types.

Associate Business Rules to Rule Description Language

Import and Import Interval business rules, as well as Interval business rules that will perform validations, must be associated to Rule Description Language (RDL) records. This association is how the Adapter knows which RDL to apply when executing a specific business rule.

There is a one-to-one relationship between RDLs and Business Rules: Only one RDL can be associated to a business rule. This means that you must create a separate RDL for each business rule you wish to run.

Create a record in the Business Rule to RDL Association table for each business rule that will have an associated RDL. This includes all Import and Import Interval rules, as well as any Interval rules for which you've configured an RDL.

Records in the Business Rule to RDL Association table contain the following fields:

- **Business Rule:** The Business Rule.
- **RDL Name:** The Rule Definition Language (RDL) name associated with the Business rule.

See **Associating Business Rules to Rule Description Language** on page 11-66 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about associating business rules to Rule Description Language.

Configuring Adapter Services

After configuring the Adapter business rules that will address the business requirements you defined, the next step is to configure the Adapter services that will run the business rules. Configuring services involves the following:

- **Create and Configure XSLT Maps and Records**
- **Create and Configure Runtime Services**
- **Create Service Activation Records**
- **Create Web Service Records**

Create and Configure XSLT Maps and Records

If your implementation will use XSLT maps for converting delimited data into XML format, you need to create the XML documents (*.xml or *.xslt) that will convert the delimited data into XML. In addition, you may also create **XSLT Map Names**, **XSLT Map Versions**, and **XSLT Maps** for each map you defined (see **Data Conversion** on page 6-3) for your implementation.

XSLT Map Names

XSLT Map Names are names and descriptions of XML stylesheets (or maps) used by the Adapter. Records in the XSLT Map Names table contain the following fields:

- **Name:** The name of the XSLT map name.
- **Description:** A description of the XSLT map name.

See **XSLT Map Names** on page 11-22 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about setting up XSLT map name records.

XSLT Map Versions

XSLT Map Versions represent specific versions of XML stylesheets (or maps) used by the Adapter. Records in the XSLT Map Versions table contain the following fields:

- **XSLT Map Name:** The XSLT Map Name record associated with the XSLT map version.
- **Description:** A description of the XSLT map version.
- **Major Version:** Major version number for the XSLT map version.
- **Minor Version:** Major version number for the XSLT map version.
- **Begin Date:** The date and time at which the XSLT map version goes into effect.
- **End Date:** The date and time after which the XSLT map version is no longer in effect.

See **XSLT Map Versions** on page 11-23 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about setting up XSLT map version records.

XSLT Maps

XSLT Maps are records which store the actual XML stylesheets (or maps) used by the Adapter. Records in the XSLT Maps table contain the following fields:

- **XSLT Map Version:** The XSLT Map Version record associated with the XSLT map.
- **XML Map:** The XML map.

See **XSLT Maps** on page 11-25 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about setting up XSLT map records.

Create and Configure Runtime Services

Runtime Services represent individual Adapter services. These are the services that run on Adapter servers which execute the business rules (and associated RDLs).

Create a record in the Runtime Service table for each service you defined for your configuration (see **Defining Business Requirements** on page 6-2 and **Runtime Service Types** on page 6-4).

There is a one-to-one relationship between Business Rules and Runtime Services. This means that you must create a separate runtime service for each business rule you wish to run. Records in the Runtime Services table contain the following fields:

- **Name:** The identifier for the runtime service. **Note:** For runtime services of type “FPORTAL” (file portal), the name of the runtime service is used to create a working directory on the file system and must not contain any of the following characters: / \ : * ? " < > |.
- **JMS Queues:** The JMS queue that serves as input for the service (if applicable).
- **JMS Filter:** Filtering syntax used to limit the JMS messages that will be available to the service.
- **JVM Parameters:** Additional parameters used by the JVM for the individual service.
- **Origination Systems:** The type of system the service is associated with.
- **Parameters:** Additional command line parameters to be applied to the service.
- **Internal Flag:** A flag that indicates (Y or N) whether or not the service is an “internal system.”
- **Output Flag:** A flag that indicates (Y or N) whether or not the service produces output.
- **Runtime Service Type:** The type of service. This can be one of the following:
 - **FPORTAL:** The Adapter File Loader Service. This service loads payloads from a file in a specified directory.
Note: Runtime services of type FPORTAL (file) that monitor a network location cannot be run on more than one application server at the same time.
 - **QPORTAL:** The Adapter Queue Loader Service. This service loads payloads from a specified JMS queue.
 - **DPORTAL:** The Adapter Database Loader Service. This service loads payloads from the Payload Table in the Oracle Utilities Data Repository.
 - **MDMPORTAL:** The MDM Web Transactions portal. This service executes operations performed by Oracle Utilities Meter Data Management users working with the application’s web-based user interface.
Note: At least one service of type MDMPORTAL must be running in order for users to use the Oracle Utilities Meter Data Management application via the web-based user interface.
Note: MDMPORTAL services are used ONLY by Oracle Utilities Meter Data Management.

See **Runtime Service Types** on page 6-4 for more information about service types.

- **Properties:** Properties associated with the service. See **Service Properties** on page 6-19 for more information about Runtime Service Properties.

See **Runtime Services** on page 11-16 in the *Oracle Utilities Energy Information Platform User’s Guide* for more information about setting up runtime service records.

Service Properties

Service Properties define how Adapter services function during runtime. Service properties are used to associate a business rule to a service property, to specify the data source(s) that the service will interact with, to specify how the service creates output based on exceptions, and other properties.

Though most services share many properties, the specific properties for a given service are largely determined by the Runtime Service Type (see **Runtime Service Types** on page 6-4). In other words, the specific properties used when implementing a service of one type (such as File/FPORTAL) can differ significantly from implementing a service of another type (such as Queue/QPORTAL or Database/DPORTAL).

General Properties: General properties define general attributes of the service. These include:

- How much data will be stored in the database,
- The character to use as a delimiter when using DataConverters (if applicable),
- The maximum number of exceptions logged to the Inbound Exception table,
- The Data Converter(s) to use in the data input phase (if applicable)
- The name of the Business Rule that should be run, and
- How much memory should be allocated to the incoming payload prior to RDL processing.

Data Source Properties: Data source properties define the data source(s) used by the Runtime Service, including the default data source, as well as optional RDL-specific data sources. These include:

- The DSN, User ID, Password, and Qualifier of the default data source
- The DSN, User ID, Password, and Qualifier of the data source that the RDL engine should use
- The DSN, User ID, Password, and Qualifier of the secondary data source that the RDL Engine should use. This can provide an alternate location for RDL rules.

Output Properties: Output properties define how services create output, including data converters and details of the output mechanism used by the service. These include:

- The Data Converter(s) to use in the data output phase for data (non-exceptions),
- The mechanism for writing outgoing data to an external target (“F” (File), “Q” (Queue), or “D” (Database)),
- The directory, file prefix, and file suffix for file-based output,
- The queue and message type for queue-based output,
- The payload type for database-based output, and
- Nodesets into which the Adapter can split payloads.

Exception Output Properties: Exception output properties define how services generate exception output, including data converters and details of the exception output mechanism used by the service.

- The Data Converter(s) to use in the data output phase for exceptions,
- The mechanism for writing exceptions to an external target (“F” (File), “Q” (Queue), or “D” (Database)),
- The directory, file prefix, and file suffix for file-based exception output,
- The queue and message type for queue-based exception output,
- The payload type for database-based exception output, and
- Nodesets into which the Adapter can split payloads.

Work Queue Properties: Work Queue properties allow the service and RDL Engine to use the Work Queues for exceptions, including the Work Queue Type, Work Queue, and work queue data source information.

- The Work Queue Type for creating Work Queue items for exceptions from the RDL Engine,
- Optional Work Queue, Priority Level, Work By Hours, Assigned to User ID, Approval Step Number, Approval Step count, and Approval Level attributes for work queue items created by exceptions from the RDL Engine, and
- The DSN, User ID, Password, and Qualifier of the data source that should be used by the Work Queues. This allows Work Queue items to be directed to a different data source.

File Portal Properties: File Portal properties are used by services with a Service Type Code of “FPORTAL” (File), and define the location of and types of files the service will pick up for processing. These include:

- The directory polled by the file service,
- The file extension the file service looks for in the specified directory, and
- The number of seconds between polling cycles

Note: Runtime services of type FPORTAL (file) that monitor a network location cannot be run on more than one application server at the same time.

Database Portal Properties: Database Portal properties are used by services with a Service Type Code of “DPORTAL” (Database), define the location of and types of payloads the service will pick up for processing. These include:

- The Payload Type and format (character or binary) of the payload to be processed (see **Data Formats** on page 6-3 and **Payload Types** on page 6-10 for more information),
- Optional database type, database driver, User ID, and password to connect to the database where the Ice Payload/Payload Extension table to be polled is located, and
- The number of seconds between polling cycles, and the number of records to luck during each polling cycle.

MDM Web Transaction Properties: MDM Web Transaction Portal properties are used by services with a Service Type Code of “MDMPORTAL” and define how the service will run. These include:

- The number of seconds between polling cycles (the default is 2 seconds for services of this type)

Note: At least one service of type MDMPORTAL must be running in order for users to use the Meter Data Management application via the web-based user interface.

Note: MDMPORTAL services are used ONLY by Oracle Utilities Meter Data Management.

See **Defining Service Properties for Adapter Runtime Services** on page 11-31 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about runtime service properties.

Create Service Activation Records

Service Activation records define how individual Adapter services will run on each server. Service Activation records link specific **Servers** to **Create and Configure Runtime Services**.

Create a Service Activation record for each Runtime Service you created above. Records in the Service Activation table contain the following fields:

- **Server:** The Server on which the service will run.
- **Runtime Services:** The service to be run.

- **Enabled:** A flag that indicates (Yes or No) if the service is allowed to run on the server.
- **Run at Startup:** A flag that indicates (Yes or No) if the service will start as soon as the server starts.
- **Keep Alive:** A flag that indicates (Yes or No) whether or not the service will restart itself in the event it shuts down.

See **Service Activation** on page 11-20 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about setting up service activation records.

Create Web Service Records

Web Services represent individual web services. Like Runtime Services, these are the services that run on Adapter servers which execute Runtime Services and Business Rules (and associated RDLs).

Records in the Web Services table contain the following fields:

- **Web Service ID:** The ID of the web service. This is used when calling the web service.
- **Description:** A description of the web service
- **Web Service Type:** The type of web service, from the Web Service Type table. Web service types include ABR (Adapter Business Rule) and ARS (Adapter Runtime Service).
- **Runtime Service:** An optional Runtime Service (from the Runtime Service table) associated with the web service
- **Business Rule:** An optional Business Rule (from the Business Rule table) associated with the web service

Note: Business Rules of type “Rate” should not be used with web services.

Using Adapter Data Converters

The Adapter includes the ability to convert data from a delimited format into an XML format that can be mapped to tables and columns in the Oracle Utilities Data Repository. This section includes descriptions of these converters and how to use them with business rules and services run by the Adapter, including:

- **CSV Parsers**
- **XML Converters**
- **Interval Data Converters**
- **Payload Splitter**
- **ZIP Converters**
- **Using Data Converters with Adapter Services**
- **Creating Custom Converters**

CSV Parsers

CSV Parsers are Java classes that parse delimited data into an XML format. The **com.lodestarcorp.core.xml.parsers** Java package contains the classes available to the Adapter for reading and parsing delimited data.

Available CSV Parsers

The Adapter includes several CSV parsers, each producing slightly different XML formats. The provided parsers include:

- **BasicCSVParser**
- **ColumnCountCSVParser**
- **RowIdentifyingCSVParser**
- **RowColumnNamingCSVParser**
- **AdaptiveCSVParser**
- **GroupingCSVParser**
- **FixedWidthConverter**
- **CSVParser**
- **XMLNester**

BasicCSVParser

The **BasicCSVParser** provides generic and simple XML. The document root is a <csv> node. Rows are contained in <row> nodes and each column is given a <col> node.

Arguments: The BasicCSVParser accepts the following arguments

Argument	Description
delimiter	The separator used in the incoming data. The default is a comma.

Argument	Description
escape	The escape sequence used in the incoming data. Escape sequences allow the delimiter to appear in the actual data. For instance, if the delimiter is a comma and the data contains a number formatted with thousands separators, an escape would be required to read the data properly: "10,000","11,000","1,000",abc. By default, the escape is a double-quote.
groupingSize	An integer that controls the number of <row> nodes collected as siblings.

Example: The following data:

```
abc,def
123,456
```

would be parsed as:

```
<csv>
  <row>
    <col>abc</col>
    <col>def</col>
  </row>
  <row>
    <col>123</col>
    <col>456</col>
  </row>
</csv>
```

ColumnCountCSVParser

The **ColumnCountCSVParser** enumerates the columns as <col_0>, <col_1>, <col_2>, etc. This can facilitate maps and easier XPath expressions.

Arguments: The ColumnCountCSVParser accepts the following arguments

Argument	Description
delimiter	The separator used in the incoming data. The default is a comma.
escape	The escape sequence used in the incoming data. Escape sequences allow the delimiter to appear in the actual data. For instance, if the delimiter is a comma and the data contains a number formatted with thousands separators, an escape would be required to read the data properly: "10,000","11,000","1,000",abc. By default, the escape is a double-quote.
groupingSize	An integer that controls the number of <row> nodes collected as siblings.

Example: The following data:

```
abc,def
123,456
```

would be parsed as:

```
<csv>
  <row>
    <col_0>abc</col_0>
    <col_1>def</col_1>
  </row>
  <row>
    <col_0>123</col_0>
    <col_1>456</col_1>
  </row>
</csv>
```

RowIdentifyingCSVParser

For even more uniqueness among node names, the **RowIdentifyingCSVParser** provides similar output as the ColumnCountCSVParser, except that the data in the first column of each row is incorporated into the row name.

Arguments: The RowIdentifyingCSVParser accepts the following arguments

Argument	Description
delimiter	The separator used in the incoming data. The default is a comma.
escape	The escape sequence used in the incoming data. Escape sequences allow the delimiter to appear in the actual data. For instance, if the delimiter is a comma and the data contains a number formatted with thousands separators, an escape would be required to read the data properly: "10,000","11,000","1,000",abc. By default, the escape is a double-quote.
groupingSize	An integer that controls the number of <row> nodes collected as siblings.

Example: The following data:

```
100,abc,def,
200,123,456,
```

would be parsed as:

```
<csv>
<row_100>
  <col_0>100</col_0>
  <col_1>abc</col_1>
  <col_2>def</col_2>
  <col_3/>
</row_100>
<row_200>
  <col_0>200</col_1>
  <col_1>123</col_1>
  <col_2>456</col_2>
  <col_3/>
</row_200>
</csv>
```

Note that because the data is incorporated into the element name, naming rules for XML nodes are implied by the data itself, rather than a schema or document type definition.

RowColumnNamingCSVParser

The **RowColumnNamingCSVParser** operates similarly to the **RowIdentifyingCSVParser**, except that the name of the row is prepended to each column's name.

Arguments: The **RowColumnNamingCSVParser** accepts the following arguments

Argument	Description
delimiter	The separator used in the incoming data. The default is a comma.
escape	The escape sequence used in the incoming data. Escape sequences allow the delimiter to appear in the actual data. For instance, if the delimiter is a comma and the data contains a number formatted with thousands separators, an escape would be required to read the data properly: "10,000","11,000","1,000",abc. By default, the escape is a double-quote.
groupingSize	An integer that controls the number of <row> nodes collected as siblings.

Example: The following data:

```
100,abc,def,
200,123,456
```

would be parsed as:

```
<csv>
<row_100>
  <row_100_col_0>abc</row_100_col_0>
  <row_100_col_1>def</row_100_col_1>
  <row_100_col_2/>
</row_100>
<row_200>
  <row_200_col_0>123</row_200_col_0>
  <row_200_col_1>456</row_200_col_1>
  <row_200_col_2/>
</row_200>
</csv>
```

AdaptiveCSVParser

The **AdaptiveCSVParser** converts separated text into XML. The document root is a <csv> node. Other nodes are created based on the value of the first column within each row. If a value has not been encountered before, it is nested under the previous node. If it has been encountered, closing elements are created for each node until the repetition is found. At that point the new row is inserted as a sibling to the previous similar value. If the last row of the document is unique, it is considered a trailing node and is appended as a child of the root node.

Arguments: The **AdaptiveCSVParser** accepts the following arguments

Argument	Description
delimiter	The separator used in the incoming data. The default is a comma.

Argument	Description
escape	The escape sequence used in the incoming data. Escape sequences allow the delimiter to appear in the actual data. For instance, if the delimiter is a comma and the data contains a number formatted with thousands separators, an escape would be required to read the data properly: "10,000","11,000","1,000",abc. By default, the escape is a double-quote.
rowOutputMode	An integer that controls the type of output format. If zero or unspecified, columns of delimited text will become attributes of row elements named col_1, col_2, col_3, ...col_n. If set to one, an element will be created with the same name as the value in the first column of separated text (note that this implies that it must be valid for xml). Other columns are specified as col_n attributes. This parameter can also accept the fully qualified class name of a Java class implementing the com.lodestarcorp.core.xml.parsers.csv.RowOutputFormatter interface.

GroupingCSVParser

The **GroupingCSVParser** behaves similarly to a BasicCSVParser, but when it notices consecutive rows with the same first column, it inserts a <set> element to contain them.

Arguments: The GroupingCSVParser accepts the following arguments

Argument	Description
delimiter	The separator used in the incoming data. The default is a comma.
escape	The escape sequence used in the incoming data. Escape sequences allow the delimiter to appear in the actual data. For instance, if the delimiter is a comma and the data contains a number formatted with thousands separators, an escape would be required to read the data properly: "10,000","11,000","1,000",abc. By default, the escape is a double-quote.

FixedWidthConverter

The **FixedWidthConverter** interprets a fixed-width payload and transforms it into xml. This converter uses a configuration file called templates.xml. This file defines, fields, their positions, and their types in an Input section. In the Output section, the xml output is described with tokens showing the location and format of the defined fields.

Arguments: The FixedWidthConverter accepts the following arguments

Argument	Description
templateName	The template.xml file can provide the instructions for multiple parsers. The name tells the dataconverter which template to use. See Templates Configuration File on page 6-28 for more information about this file.

CSVParser

The **CSVParser** interprets a CSV payload and transforms it into xml. This converter uses a configuration file called templates.xml. This file defines fields, their column numbers, and their types an Input section. In the Output section, the xml output is described with tokens showing the location and format of the defined fields.

Arguments: The CSVParser accepts the following arguments

Argument	Description
templateName	The template.xml file can provide the instructions for multiple parsers. The name tells the dataconverter which template to use. See Templates Configuration File on page 6-28 for more information about this file.
delimiter	The separator used in the incoming data. The default is a comma.
escape	The escape sequence used in the incoming data. Escape sequences allow the delimiter to appear in the actual data. For instance, if the delimiter is a comma and the data contains a number formatted with thousands separators, an escape would be required to read the data properly: "10,000","11,000","1,000",abc. By default, the escape is a double-quote.

XMLNester

The XMLNester converter inserts a grouping node in CSV-parsed XML. All CSV parsers create a root node of <csv>. This parser groups children of that node within a node specified by the user. The extra level can make splitting and XSLT easier.

Arguments: The XMLNester accepts the following arguments

Argument	Description
groupSize	The number of children to collect into each group. The default value is 500.

Argument	Description
groupName	The name of the grouping node. By default, it is named "group."

Using CSV Parsers with Adapter Services

To use one of the above CSV parsers with an Adapter service, you must specify the parser in one of the appropriate Service Properties. These include:

- Input Data Converter (**INPUT_DC_n**)
- Output Data Converter (**OUTPUT_DC_n**)
- Exception Output Data Converter (**EXCEPTION_OUTPUT_DC_n**)

See **Defining Service Properties for Adapter Runtime Services** on page 11-31 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about service properties.

Format

The format for specifying parsers is:

```
<PROPERTY NAME="INPUT_DC_0"
VALUE="com.lodestarcorp.core.xml.parsers.<PARSER_NAME>,<ARGUMENTS>" />
```

where:

- **<PARSER_NAME>** is the name of the parser you wish to specify.
- **<ARGUMENTS>** are optional arguments used by the parser. Arguments are optional, but arguments cannot be skipped. For example, if a dataconverter has four arguments, the first two must be supplied in order to specify the third.

Example:

Use the "ColumnCountCSVParser" as the CSV parser.

```
<PROPERTY NAME="INPUT_DC_0"
VALUE="com.lodestarcorp.core.xml.parsers.ColumnCountCSVParser" />
```

Specifying Delimiters

By default, the parsers use the comma “,” character as the delimiter, but they can also process data delimited with other characters. To use a different character as the delimiter, specify the delimiter character as the first argument to the converter. Use “\t” to specify the Tab character.

See **Defining Service Properties for Adapter Runtime Services** on page 11-31 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about service properties.

Examples:

Use the “|” character as the delimiter:

```
<PROPERTY NAME="INPUT_DC_0"
VALUE="com.lodestarcorp.core.xml.parsers.ColumnCountCSVParser,|" />
```

Use the Tab character as the delimiter:

```
<PROPERTY NAME="INPUT_DC_0"
VALUE="com.lodestarcorp.core.xml.parsers.ColumnCountCSVParser,\t" />
```

Templates Configuration File

The CSVParser and FixedWidthConverter parsers both use a configuration file called Templates.xml to define the input format accepted by the converters as well as the output format the converters produce.

Templates.xml - Sample

```
<fwc:Templates xmlns="http://www.lodestarcorp.com/target"
xmlns:fwc="http://www.lodestarcorp.com/fixedwidthconverter">
  <fwc:Template name="SpecificFileType" version="1.0">
    <fwc:Input>
      <fwc:Fields>
        <fwc:Field id="meterID" type="string" start="22" size="21"/>
        <fwc:Field id="readValue" type="decimal" start="70" size="9"/>
        <fwc:field id="stopDate" type="date" start="90" size="10"
          inputFormat="MM/dd/yyyy&apos;T&apos;"/>
        <fwc:field id="stopTime" type="date" start="100" size="8"
          inputFormat="HH:mm:ss"/>
      </fwc:Fields>
    </fwc:Input>
    <fwc:Output>
      <LODESTARUSAGE>
        <MDMUSAGE fwc:record="true" CHANNELID="01" UOM="01">
          <fwc:Attributes>
            <fwc:Attribute id="METERID" value="meterID"/>
          </fwc:Attributes>
          <METERREADS>
            <METERREAD>
              <fwc:Attributes>
                <fwc:Attribute name="JURISCODE" value="$JURISCODE"/>
                <fwc:AttributeGroup id="STOPREADTIME">
                  <fwc:Attribute value="stopDate"/>
                  <fwc:Attribute value="stopTime"/>
                </fwc:AttributeGroup>
                <fwc:Attribute id="STOPVAL" value="readValue"/>
              </fwc:Attributes>
            </METERREAD>
          </METERREADS>
        </MDMUSAGE>
      </LODESTARUSAGE>
    </fwc:Output>
  </fwc:Template>
</fwc:Templates>
```

Templates.xml - Elements and Attributes

The Templates.xml file contains the following elements and attributes:

fwc:Templates: Root element that contains one or more fwc:Template elements, each defining a specific template.

Attributes:

- **xmlns:** The XML namespace for the output XML produced by the converter.
- **xmlns:fwc:** The XML namespace for all template elements.

Elements:

- **fwc:Template:** Element that defines a specific template used by either the FixedWidthCoverter or CSVParser converter.

Attributes:

- **Name:** The name of the template, used by Adapter Runtime Services to select the proper configuration for input and output. The template name used by a converter is defined as the “templateName” argument when specifying the converter in the Runtime Service properties.
- **Version:** The version of the template. Should always be “1.”

Elements:

- **fwc:Input:** Element that defines input format for the template.

Elements:

- **fwc:Fields:** Element containing one or more fwc:Field elements, each defining a specific field

Elements:

- **fwc:Field:** Element defining a single field in the input format.

Attributes:

- **id:** Id for the field. Defined using camel case
- **type:** The type of field. Valid types include “string”, “decimal”, and “date” (used for dates, times, and datetimes)
- **start:** The starting location of the field in the input format
- **size:** The length of a piece of a fixed width line that corresponds to the field
- **decimals:** The number of digits to the right of the decimal point the field value should have. Used with fields of type “decimal” only.
- **inputformat:** Input format for date fields. Can be any format that defines dates and/or times, where “MM” = Month, “dd” = Day, “yy” or “yyyy” = Year, “HH” = Hour, “mm” = Minutes, “ss” = seconds.
- **outputformat:** Optional output format for date fields (see inputformat above for formatting options)

- **fwc:Output:** Element that defines output format for the template.

Elements:

- **<ELEMENT>:** User-defined root element for output. Should contain child elements that define the format of the output.

Attributes:

- **fwc:record:** Specifies whether or not the element represents a single record of data. In most cases, a record is defined by a single line of fixed width data. The "record" instruction tells the parser to repeat that node for each line of data. In this way, the values assigned to the Fields will be recomputed for each line.

Elements:

- **fwc:Attributes:** Element containing one or more fwc:Attribute elements, each defining a specific attribute of the parent element. Used to create one or more attributes within the parent output element.

Elements:

- **fwc:Attribute:** Element defining a single attribute within an element. Used to an attribute within the parent output element

Attributes:

- **name:** The name of an attribute to be inserted into the parent output element. Used ONLY when passing system variables to the output. The format for system variables is “\$<name>” where <name> is the name of the system variable.
- **id:** Optional ID of the attribute to be inserted into the parent output element. If not included, the name of the attribute defined in the “value” attribute is used as the attribute ID.
- **value:** The ID of the input field used for the attribute value.
- **fwc:AttributeGroup:** Used to concatenate fields from the input defined in fwc:Attribute elements into a single output field.

XML Converters

XML converters are Java classes that convert XML data and validate XML data. The **com.lodestarcorp.portal.data** Java package contains the classes available to the Adapter.

Available XML Converters

The Adapter includes two XML converters including:

- **XSLConverter**
- **SchemaValidator**

XSLConverter

The XSLConverter converts delimited data based on a specified XSLT map. This map can be either a file on the Adapter server (or network) or a XSLT Map stored in the XSLT Maps table in the Oracle Utilities Data Repository.

SchemaValidator

The SchemaValidator validates XML data against a specified XML schema (*.xsd). The XML data to be validated can be either incoming data in XML format, or data that has been previously converted via XSLT or one of the CSV parsers.

Using XML Converters with Adapter Services

To use one of the above XML converters with an Adapter service, you must specify the converter in one of the appropriate Service Properties. These include:

- Input Data Converter (**INPUT_DC_n**)
- Output Data Converter (**OUTPUT_DC_n**)
- Exception Output Data Converter (**EXCEPTION_OUTPUT_DC_n**)

See **Defining Service Properties for Adapter Runtime Services** on page 11-31 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about service properties.

Format

The format for specifying converters is:

```
<PROPERTY NAME="INPUT_DC_0"
VALUE="com.lodestarcorp.portal.data.<CONVERTER_NAME>,<XML_NAME>" />
```

where:

- **<CONVERTER_NAME>** is the name of the converter you wish to use, either "XMLCoverter" or "SchemaValidator."
- **<XML_NAME>** is the name of either the XSLT map (*.xslt) used for converting data, or the XML schema (*.xsd) used to validate the XML data.
 - When specifying files (*.xslt or *.xsd), you must include the path and file name of the file.
 - When specifying XSLT maps stored in the Oracle Utilities Data Repository, use the following convention:

```
com.lodestarcorp.portal.data.XSLConverter,DC://<XML_NAME>
```

When using XSLT maps stored in the Oracle Utilities Data Repository, the most recent version (based on the XSLT Map Versions table) will be used.

Examples:

Use the XSLConverter with the “CSVMap.xslt” file.

```
<PROPERTY NAME="INPUT_DC_0"
VALUE="com.lodestarcorp.portal.data.XSLConverter,C:\Interval\CSVMap.xslt"/>
```

Use the XSLConverter with the “Meter_CSVMap” stored in the Oracle Utilities Data Repository.

```
<PROPERTY NAME="INPUT_DC_0"
VALUE="com.lodestarcorp.portal.data.XSLConverter,DB://Meter_CSVMap"/>
```

Using the SchemaValidator using the “IUsage.xsd” schema.

```
<PROPERTY NAME="INPUT_DC_0"
VALUE="com.lodestarcorp.portal.data.SchemaValidator,C:\Interval\IUsage.xsd"/>
```

Interval Data Converters

Interval data converters are Java classes that convert interval data into an XML format. The **com.lodestarcorp.portal.data** Java package contains the classes available to the Adapter.

Available Interval Data Converters

The Adapter includes two interval data converters including:

- **LSEParser**
- **IntDExpConverter**
- **MV90Converter (Used with Oracle Utilities Meter Data Management Only)**
- **MV9Converter (Used with Oracle Utilities Meter Data Management Only)**
- **LSEConverter (Used with Oracle Utilities Meter Data Management Only)**
- **LegacyMV90Converter (Used with Oracle Utilities Meter Data Management Only)**

LSEParser

The LSEParser converts an incoming LSE-formatted payload into XML. Once in XML format, large LSE payloads can be broken apart by the Payload Splitter and imported separately.

Arguments: The LSEParser accepts the following arguments

Argument	Description
groupedReads	Counts the number of lines starting with "00000001." When the count reaches the value in this parameter, the dataconverter creates an intermediate node named READGROUP with a COUNT attribute. The default value is 500.

IntDExpConverter

The IntDExp Converter is a Java class that converts interval data into an XML format that can be mapped into the Oracle Utilities Data Repository using Rule Description Language. The **com.lodestarcorp.portal.data** Java package contains this class (IntDExpConverter).

The IntDExp Converter converts interval data in any of the Oracle Utilities supported interval data formats, including:

- Oracle Utilities Enhanced Database Format (Pervasive.SQL) (*.bte)
- Oracle Utilities Enhanced Input/Output Interval Data Format (*.lse)
- Oracle Utilities Standard Interval Data Format (*.inp)

- Oracle Utilities Comma Separated Interval Data Format (*.csv)
- Oracle Utilities Standard XML Interval Data Format (*.xml)
- MV90 Mainframe Format (*.mv9)
- EU Engineering Units Format (*.eu)
- DTV Engineering Units Format (*.dtv)
- Meter Data Exchange Format (*.mdf)

See **Appendix C: Oracle Utilities Enhanced Input/Output Interval Data Format**, **Appendix D: Oracle Utilities Standard XML Interval Data Format**, and **Appendix E: Oracle Utilities Comma Separated Values (CSV) Interval Data Format** in the *Oracle Utilities Energy Information Platform Configuration Guide* for more information about these interval data formats.

Arguments: The IntDExpConverter accepts the following arguments

Argument	Description
operatingExtension	The file extension for the expected type of payload. Options could include mv9, lse, and xml. The default is mv9.
debugging	This true or false parameter enables or disables debugging mode. In normal operation this parameter would be set this to false, but when true the temporary files are not deleted. The default value is false.
configurationPath	When provided, this argument is used to specify the location of the lodestar.cfg file which can be used to configure certain properties of intdexp.exe. By default, this argument is unused.

MV90Converter (Used with Oracle Utilities Meter Data Management Only)

The MV90Converter is a Java class that converts interval data into the Oracle Utilities Meter Data (LSM) format used by the Oracle Utilities Meter Data Management application. The **com.lodestarcorp.portal.data** Java package contains this class (MV90Converter). See **MV90 Interval Data Converter** on page 5-21 in the *Oracle Utilities Meter Data Management Installation and Configuration Guide* for more information about this converter.

MV9Converter (Used with Oracle Utilities Meter Data Management Only)

The MV9 Converter is a Java class that converts interval data in the MV90 format into the Oracle Utilities Meter Data (LSM) format used by the Oracle Utilities Meter Data Management application. The **com.lodestarcorp.portal.data.intd** Java package contains this class (MV9Converter). See **MV90 Interval Data Converter** on page 5-21 in the *Oracle Utilities Meter Data Management Installation and Configuration Guide* for more information about this converter. See **MV9 Interval Data Converter** on page 5-23 in the *Oracle Utilities Meter Data Management Installation and Configuration Guide* for more information about this converter.

LSEConverter (Used with Oracle Utilities Meter Data Management Only)

The LSE Converter is a Java class that converts interval data in the Oracle Utilities Enhanced Input/Output Interval Data Format (*.lse) format into the Oracle Utilities Meter Data (LSM) format used by the Oracle Utilities Meter Data Management application. The **com.lodestarcorp.portal.data.intd** Java package contains this class (LSEConverter). See **LSE Interval Data Converter** on page 5-24 in the *Oracle Utilities Meter Data Management Installation and Configuration Guide* for more information about this converter.

LegacyMV90Converter (Used with Oracle Utilities Meter Data Management Only)

The LegacyMV90 Converter is a Java class that converts interval data into the Oracle Utilities Meter Data (LSM) format used by the Oracle Utilities Meter Data Management application. The `com.lodestarcorp.portal.data` Java package contains this class (LegacyMV90Converter). See **LegacyMV90 Interval Data Converter** on page 5-25 in the *Oracle Utilities Meter Data Management Installation and Configuration Guide* for more information about this converter.

Using Interval Data Converters with Adapter Services

To use one of the above interval data converters with an Adapter service, you must specify the converter in one of the appropriate Service Properties. These include:

- Input Data Converter (**INPUT_DC_n**)

See **Defining Service Properties for Adapter Runtime Services** on page 11-31 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about service properties.

Format

The format for specifying interval data converters is:

```
<PROPERTY NAME="INPUT_DC_0"  
VALUE="com.lodestarcorp.portal.data.<CONVERTER>[, <FILE_EXTENSION>,  
<ARGUMENTS>]" />
```

where:

- **<CONVERTER>** is the name of the converter you wish to specify.
- **<FILE_EXTENSION>** is the name of an optional file format extension (lse, mdf, etc.) for the data to be converted when using the IntDExp converter. The default file format is MV90 (mv9), and need not be specified when converting data from MV90 format.
 - When specifying a file format, you must include the comma between the classname (MV90Converter) and the file format extension. If not specifying a file format, the comma should NOT be included.
- **<ARGUMENTS>** are optional arguments used by the converter. Arguments are optional, but arguments cannot be skipped. For example, if a dataconverter has four arguments, the first two must be supplied in order to specify the third.

Examples:

Use the "LSEParser" to convert incoming interval data in LSE format.

```
<PROPERTY NAME="INPUT_DC_0" VALUE="com.lodestarcorp.portal.data.LSEParser" />
```

Use the IntDExp Converter to convert an MV90.

```
<PROPERTY NAME="INPUT_DC_0"  
VALUE="com.lodestarcorp.portal.data.IntDExpConverter" />
```

Use the IntDExp Converter to convert an LSE.

```
<PROPERTY NAME="INPUT_DC_0"  
VALUE="com.lodestarcorp.portal.data.IntDExpConverter,lse" />
```

Payload Splitter

By default, the Adapter uses an all-or-nothing approach when applying validations and other logic, meaning that if a single record within a payload fails a validation, the entire payload is rejected.

The Payload Splitter provides a means for breaking up XML payloads into smaller nodesets. This allows each nodeset (containing one or more records) to be validated separately.

Using the Payload Splitter means essentially dividing the work for processing an incoming payload into two services: the first splits the payload, and the second performs business logic to be applied to the payload, such as validations and data inserts/updates.

Note: The output of the Payload Splitter is performed after any other business rule processing associated with the Runtime Service. To process the payload as it is initially processed by the Adapter (i.e. before any other business rules), omit the RULENAME property with services employing the Payload Splitter.

Using the Payload Splitter with Adapter Services

To use the Payload Splitter with an Adapter service, you must specify the xpath by which the payload is to be split in one of the following Service Properties.

- Output Splitter (**OUTPUT_DIVIDING_XPATH** or **OUTPUT_DIVIDING_STRING**)
- Exception Output Splitter (**EXCEPTION_OUTPUT_DIVIDING_XPATH** or **EXCEPTION_OUTPUT_DIVIDING_STRING**)
- Output Name Count (**OUTPUT_NAME_COUNT**), used with **OUTPUT_DIVIDING_STRING**
- Output Namespaces (**OUTPUT_NAMESPACES**, see **XML Namespaces** on page 6-37 for more information about the use of this property)
- Exception Output Name Count (**EXCEPTION_OUTPUT_NAME_COUNT**) used with **EXCEPTION_OUTPUT_DIVIDING_STRING**
- Exception Output Namespaces (**EXCEPTION_OUTPUT_NAMESPACES**, see **XML Namespaces** on page 6-37 for more information about the use of this property)

See **Defining Service Properties for Adapter Runtime Services** on page 11-31 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about service properties.

Format - OUTPUT_DIVIDING_XPATH

The format for specifying how to split a payload via **OUTPUT_DIVIDING_XPATH** is:

```
<PROPERTY NAME="OUTPUT_DIVIDING_XPATH" VALUE="<XML_XPATH>" />
```

where:

- **<XML_XPATH>** is the xpath that defines how to split the payload. As each node in the nodeset is processed, it is outputted via the method set in the **OUTPUT_MECHANISM** service property.

Example:

Use the splitter to convert the following XML payload.

```
<METERREADS>
  <METERREAD>
    <METERID>METER123</METERID>
    <STARTREADING/>
    ...
  </METERREAD>
  <METERREAD>
    <METERID>METER456</METERID>
    <STARTREADING/>
    ...
  </METERREAD>
</METERREADS>
```

Into nodesets that contain the METERREAD (and related child) node(s), and insert each into the Payload Extension table with a Payload Type of "METERREADS."

```
<PROPERTY NAME="OUTPUT_MECHANISM" VALUE="D" />
<PROPERTY NAME="OUTPUT_DATABASE_PAYLOADTYPE" VALUE="METERREADS" />
<PROPERTY NAME="OUTPUT_DIVIDING_XPATH" VALUE="/METERREADS/METERREAD" />
```

The output of a service with the property would be:

Record 1:

```
<METERREAD>
  <METERID>METER123</METERID>
  <STARTREADING/>
  ...
</METERREAD>
```

Record 2:

```
<METERREAD>
  <METERID>METER456</METERID>
  <STARTREADING/>
  ...
</METERREAD>
```

Format - OUTPUT_DIVIDING_STRING

The format for specifying how to split a payload via OUTPUT_DIVIDING_STRING is:

```
<PROPERTY NAME="OUTPUT_DIVIDING_STRING" VALUE="<STRING>" />
```

where:

- **<STRING>** is a series of XML element names separated by vertical pipe characters (“|”) that defines how to split the payload. As each node in the nodeset is processed, it is outputted via the method set in the OUTPUT_MECHANISM service property.

Example:

Use the splitter to convert the following XML payload.

```
<METERREADS>
  <METERREAD>
    <METERID>METER123</METERID>
    <STARTREADING/>
    ...
  </METERREAD>
  <METERREAD>
    <METERID>METER456</METERID>
    <STARTREADING/>
    ...
  </METERREAD>
</METERREADS>
```

Into nodesets that contain the METERREAD (and related child) node(s), and insert each into the Payload Extension table with a Payload Type of “METERREADS.”

```
<PROPERTY NAME="OUTPUT_MECHANISM" VALUE="D" />
<PROPERTY NAME="OUTPUT_DATABASE_PAYLOADTYPE" VALUE="METERREADS" />
<PROPERTY NAME="OUTPUT_DIVIDING_STRING" VALUE="|METERREADS|METERREAD" />
```

The output of a service with the property would be:

Record 1:

```
<METERREAD>
  <METERID>METER123</METERID>
  <STARTREADING/>
  ...
</METERREAD>
```

Record 2:

```
<METERREAD>
  <METERID>METER456</METERID>
  <STARTREADING/>
  ...
</METERREAD>
```

XML Namespaces

Sometimes incoming data exists within a specific namespace, which must be provided to the Adapter in order to properly process the data. In this instance, the `OUTPUT_NAMESPACES` property is used to define the namespaces used by the `OUTPUT_DIVIDING_XPATH` property.

Example 1:

In the following example, the xml has a default namespace (all nodes within the document exist in the “http://mycompany.com” namespace).

```
<METERS xmlns="http://mycompany.com">
  <METER id="442994">
    <READING time="2006-01-01T15:45:00">8837</READING>
  </METER>
  <METER id="442995">
    <READING time="2006-01-01T15:45:00" >8462</READING>
  </METER>
  <METER id="442996">
    <READING time="2006-01-01T15:45:00">2201</READING>
  </METER>
</METERS>
```

A valid xpath might be “/METERS/METER”. No prefix is used to refer to nodes, so the `OUTPUT_NAMESPACES` property should be set to “http://mycompany.com”

```
<PROPERTY NAME="INPUT_DC_0" VALUE="/METERS/METER"/>

<PROPERTY NAME="OUTPUT_NAMESPACES" VALUE="http://mycompany.com"/>
```

Example 2:

This is a more complicated XML. The METERS node is in the default namespace (http://mycompany.com). However, the METER nodes are in the http://anothercompany.com namespace.

```
<METERS xmlns="http://mycompany.com" xmlns:pre=http://anothercompany.com>
  <pre:METER id="442994">
    <READING time="2006-01-01T15:45:00">8837</READING>
  </pre:METER>
  <pre:METER id="442995">
    <READING time="2006-01-01T15:45:00" >8462</READING>
  </pre:METER>
  <pre:METER id="442996">
    <READING time="2006-01-01T15:45:00">2201</READING>
  </pre:METER>
</METERS>
```

To reference them in the xpath, you would use a prefix: “/METERS/pre:METER” in the `OUTPUT_DIVIDING_XPATH` property. Also, the namespaces must be defined in the `OUTPUT_NAMESPACES` property:

```
<PROPERTY NAME="INPUT_DC_0" VALUE="/METERS/pre:METER"/>

<PROPERTY NAME="OUTPUT_NAMESPACES" VALUE="http://mycompany.com pre=http://
anothercompany.com"/>
```

ZIP Converters

XIP converters are Java classes that expand/compress data in GZIP format. The `com.lodestarcorp.portal.data` Java package contains the classes available to the Adapter.

Available ZIP Converters

The Adapter includes two ZIP converters including:

- `UnzipDataConverter`
- `ZipDataConverter`

UnzipDataConverter

The `UnzipDataConverter` expands an incoming stream from GZIP format. Once the incoming data has been expanded, it can be passed to another converter or a Business Rule for processing.

ZipDataConverter

The `ZipDataConverter` compresses a payload into the GZIP format. This converter is typically used as an output converter.

Using ZIP Converters with Adapter Services

To use one of the above ZIP converters with an Adapter service, you must specify the converter in one of the appropriate Service Properties. These include:

- Input Data Converter (`INPUT_DC_n`)
- Output Data Converter (`OUTPUT_DC_n`)
- Exception Output Data Converter (`EXCEPTION_OUTPUT_DC_n`)

See **Defining Service Properties for Adapter Runtime Services** on page 11-31 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about service properties.

Format

The format for specifying ZIP converters is:

```
<PROPERTY NAME="INPUT_DC_0" VALUE="com.lodestarcorp.portal.data.<CONVERTER>" />
```

where:

- `<CONVERTER>` is the name of the parser you wish to specify.

Example:

Use the “`UnzipDataConverter`” to expand incoming data.

```
<PROPERTY NAME="INPUT_DC_0"
VALUE="com.lodestarcorp.portal.data.UnzipDataConverter" />
```

Use the “`ZipDataConverter`” to compress outgoing data.

```
<PROPERTY NAME="OUTPUT_DC_0"
VALUE="com.lodestarcorp.portal.data.ZipDataConverter" />
```


Using Data Converters with Adapter Services

Data converters can be used together to perform multiple transformations on incoming data.

Multiple data converter service properties (“INPUT_DC_n”, “OUTPUT_DC_n”, and “EXCEPTION_OUTPUT_DC_n”) can be “chained” together by listing each in a separate service property with the appropriate ordinal number appended to the property name (INPUT_DC_0, INPUT_DC_1, INPUT_DC_2, etc.). When configured in this way, conversion of the data is performed for each data converter listed, in the order listed. **Note:** When listing multiple converters, the first **must** be “0”.

For example, incoming delimited data could be converted to generic XML using one of the CSV parsers. That generic XML could then be converted into a more specific XML format using XSLT, which could then be validated against an XML schema.

Example

In the following example, incoming usage data in tab-delimited format is converted into XML using a CSV parser (“ColumnCountCSVParser”), then converted again using an XSLT map in the Oracle Utilities Data Repository (“CSVMap”), and then validated against an XML schema (“ValUsage.xsd”).

```
<PROPERTIES>
  <PROPERTY NAME="DC_DELIMITER" VALUE="\t"/>
  <PROPERTY NAME="INPUT_DC_0"
VALUE="com.lodestarcorp.core.xml.parsers.ColumnCountCSVParser"/>
  <PROPERTY NAME="INPUT_DC_1"
VALUE="com.lodestarcorp.portal.data.XSLConverter,DB://CSVMap"/>
  <PROPERTY NAME="INPUT_DC_2"
VALUE="com.lodestarcorp.portal.data.SchemaValidator,C:\Adapter\Converters\
ValUsage.xsd"/>
</PROPERTIES>
```

Creating Custom Converters

In addition to the data converters provided, you can also create custom data converters for use with your Adapter services. This section describes the processes for adding a custom converter to your implementation, and technical details concerning the basic data converter interfaces used by the **com.lodestarcorp.core.data** package.

To add a custom converter to your Adapter implementation, use the following procedure:

1. Create a .jar file that contains the Java classes for your custom converters.
2. Copy the .jar file onto each application server that will be running the Adapter.
3. Edit the Class Path value of the System Properties record for each application server.

Note: Separate the new jar's path from the other paths listed with a semicolon, as in the example (EXAMPLE.jar) below.

```
.\portal.jar;.\EXAMPLE.jar;.\ltmh.jar;.\lsice.jar;.\CustomerBusinessRu
les.jar;.\lib\jms.jar;.\lib\tibjms.jar;.\lib\com.ibm.mq.jar;.\lib\com.
ibm.mqjms.jar;.\lib\xercesImpl.jar;.\lib\xalan.jar;.\lib\xml-
apis.jar;.\lib\ojdbc14.jar;.\lib\fscontext.jar;.\lib\providerutil.jar;
.\lib\commons-cli-1.0.jar;.\lib\commons-lang-2.0.jar
```

4. Restart the Adapter server.

Basic Data Converter Interfaces

The `com.lodestarcorp.core.data` package contains the relevant classes for implementing DataConverters. The basic interfaces provided in this package are described below.

DataConsumer Interface

```
public interface DataConsumer
```

Fields

None

Methods

```
attachInputStream(LSInputStream)
```

```
attachStreamClosingListener(StreamClosingListener)
```

This is the simplest interface. Its main purpose is to consume data. Therefore, it needs a way to get the data. Nothing is implied about actually processing the data.

Methods:

attachInputStream(iStream)

```
public <M extends Metadata> boolean  
    attachInputStream(LSInputStream<M> iStream)
```

This method accepts an `LSInputStream` and returns a boolean value showing whether or not the stream was successfully attached. A `DataConverter` may choose to disallow connections at certain points in its operation. For example, if processing has already started, it probably doesn't make sense to allow the data stream to be changed. In this instance, `false` could be returned.

attachStreamClosingListener(StreamClosingListener)

```
public void  
    attachStreamClosingListener(StreamClosingListener listener)
```

Sometimes a `DataConverter` encounters some problem while processing data and needs to quit reading from the source stream. If that happens, the object writing to the stream should stop writing to the stream. Failure to stop writing could lead to errors or worse, a deadlock. `DataConsumer` implementations should signal the `StreamClosingListener` provided through this method prior to terminating in error. The signal will alert the writing object that it should cease writing immediately.

DataProcessor Interface

```
public interface DataConsumer
```

```
public interface DataProcessor extends Runnable, DataConsumer
```

Fields

RESOURCES

ALREADY_RUNNING

DISCONNECTED

STREAM_ATTACHMENT_ERROR

Methods

```
setErrorObserver(ErrorObserver)
```

Just as the `DataConsumer` interface implies only a mechanism to read data, this class adds the ability to process it. The `DataProcessor` interface inherits from two parents: `DataProcessor` and

java.lang.Runnable. The run() method of Runnable provides a method in which processing takes place. Inheriting from Runnable also provides a way to perform the processing within a dedicated thread.

Fields:

RESOURCES

```
protected static LanguageDictionary RESOURCES
```

A static instance of the com.lodestarcorp.core.util.LanguageDictionary utilities class. It is used to extract appropriate localized strings using keys. Generally, api users will not need this object.

ALREADY_RUNNING

```
protected static String ALREADY_RUNNING
```

A String containing an error message to be used when a user attempts to start processing when processing has already begun.

DISCONNECTED

```
protected static String DISCONNECTED
```

A String containing an error message indicating that the user has directed processing to begin, but either no InputStream is attached, it is null, or it is already closed.

STREAM_ATTACHMENT_ERROR

```
protected static String STREAM_ATTACHMENT_ERROR
```

A String containing an error message that can be used when the attachInputStream() method returns false. A common scenario in which the attachInputStream() method can return false is when the user attempts to re-attach an InputStream before processing has been completed.

Methods:

setErrorObserver(ErrorObserver)

```
public void setErrorObserver(ErrorObserver eo)
```

This method supports error handling when running the DataProcessor in a separate thread. In particular, when several DataConverters are running together in a chain, errors tend to cascade. That is, an error in one converter tends to cause errors in the others running concurrently. If this happens, it can be difficult to determine which error was the root cause of the problem. The ErrorObserver class provides a way to detect the first error.

DataProducer Interface

```
public interface DataProducer extends StreamClosingListener
```

Fields

None

Methods

```
attachOutputStream(InputStream)
```

This interface represents a class that can produce data. Nothing is implied about where it might acquire the data. However, as a StreamClosingListener, it is aware that there might be a mechanism reading from the output stream and it might be required to change its behavior if that mechanism stops reading the data.

Methods:**attachOutputStream(LSOutputStream)**

```
public <M extends Metadata> boolean  
    attachOutputStream(LSOutputStream<M> oStream)
```

This method takes an LSOutputStream parameter and returns a boolean value showing whether or not the method was successful. A DataConverter may choose to disallow connections at certain points in its operation. For example, if processing has already started, it probably doesn't make sense to allow the output stream to be changed.

DataConverter Interface

```
public interface DataConverter extends DataProcessor, DataProducer
```

Fields

None

Methods

None

This interface defines no new methods, but is still the most important interface in the api. It represents the union of DataProducer and DataProcessor. Conceptually, it has the ability to read data, process it, and output it.

StreamClosingListener Interface

```
public interface StreamClosingListener
```

Fields

None

Methods

closeAlert()

Classes that implement StreamClosingListener can respond to stream closing alerts when registered through the attachStreamClosingListener() method of DataConsumer.

Methods:**closeAlert()**

```
public void closeAlert()
```

A DataConsumer calls this method to alert listeners that it will read no further data from the supplied input stream. Many times a class responsible for providing data depends on this signal to avoid deadlocks. Typically, in the producer-consumer model, the producer side will block its thread if the buffer is full while it waits for the consumer to take data and make space in the buffer. If the consumer will never read the data again, it is almost guaranteed that the producer will block. This method allows the producer side to either interrupt the block or ensure that the block is never reached. Note that the signal will arrive in a different thread than the producing thread.

Implementing Adapter Services and Business Rules

The steps outlined above in the section entitled **Configuring the Adapter** provide a generic overview of how to configure the Adapter. However, there specific steps that apply to each type of business rule (Import, Import Interval, COM, etc.). This section outlines the specific steps involved in implementing business rules and services of each type.

Implementing Import Services and Business Rules

The steps outlined below should be performed for **each** Import business rule you wish to run using the Adapter

Create and Configure Import Global Functions and Versions

The first step in implementing Import services and business rules is to create and configure any Global Functions used by Import business rules.

Global Function records for Import services and business rules **MUST** include the following:

- **Business Rule Type: Import**

See **Create and Configure Global Functions** on page 6-11 for more information about creating and configuring Global Functions and Versions.

Note: This step need only be performed once, since Global Functions by definition are shared by multiple business rules.

Create RDLs and RDL Versions

The next step is to create the Rule Description Language (RDL) records and RDL Versions for each business rule.

Rule Description Language records for Import services and business rules **MUST** include the following:

- **Rule Type Code: Import**

See **Create and Configure Rule Description Language** on page 6-12 for more information about creating RDLs and RDL Versions.

Configure RDLs

The next step is to configure each RDL Version that will be used. For Import RDLs, this involves the following:

- Mapping data elements/attributes to tables/columns
- Configuring Table Validations
- Configuring Mappings
- Validating RDL

See **RDL Configuration** on page 6-13 for more information about configuring RDLs.

Create Business Rule Record

The next step is to create the Business Rule records that represent the specific business rules.

Business Rule records for Import services and business rules **MUST** include the following:

- **Rule Type: Import**

See **Create and Configure Business Rule Records** on page 6-15 for more information about creating Business Rule records.

Define Import Business Rule Properties and Parameters

As part of creating the Business Rule record, you must also define the Business Rule properties and parameters

Import business rules use the following properties:

- **xmlstream:** Instructs the Adapter parsing engine to keep the XML data that is to be sent to the database engine
- **mode:** Instructs the Adapter parsing engine to perform database operations on a per table basis (“event”) or as a batch done at the end of the incoming payload document (“batch”).

Example:

```
<PROPERTIES>
  <PROPERTY NAME='xmlstream' VALUE='y' />
  <PROPERTY NAME='mode' VALUE='event' />
</PROPERTIES>
```

See **Defining Properties and Parameters for Adapter Business Rules** on page 11-114 and **Import Properties and Parameters** on page 11-115 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about business rule properties and parameters.

Associate Business Rule to RDL

The next step is to associate each Business Rule record to the corresponding RDL. See **Associating Business Rules to Rule Description Language** on page 11-66 in the *Oracle Utilities Energy Information Platform User's Guide* for more information.

Optional: Create XSLT Map Names, Map Versions, and Maps

If the business rule and service will use XSLT maps for transforming data, you must also create XSLT Map Name records, XSLT Map Versions, and XSLT Maps. See **Create and Configure XSLT Maps and Records** on page 6-17 for more information.

Create Runtime Service

The next step is to create a Runtime Service record for each business rule. See **Create and Configure Runtime Services** on page 6-18 for more information.

Define Service Properties

As part of creating the Runtime Service record, you must also define the Service properties. See **Service Properties** on page 6-19 and **Defining Service Properties for Adapter Runtime Services** on page 11-31 in the *Oracle Utilities Energy Information Platform User's Guide* for more information.

Create Service Activation record

The last step is to create a Service Activation record for each Runtime Service. See **Create Service Activation Records** on page 6-20 for more information.

Implementing Import Interval Services and Business Rules

The steps outlined below should be performed for **each** Import Interval business rule you wish to run using the Adapter.

Create and Configure Import Interval Global Functions and Versions

The first step in implementing Import Interval services and business rules is to create and configure any Global Functions used by Import Interval business rules.

Global Function records for Import Interval services and business rules **MUST** include the following:

- **Business Rule Type: Import Interval**

See **Create and Configure Global Functions** on page 6-11 for more information about creating and configuring Global Functions and Versions.

Note: This step need only be performed once, since Global Functions by definition are shared by multiple business rules.

Create RDLs and RDL Versions

The next step is to create the Rule Description Language (RDL) records and RDL Versions for each business rule.

Rule Description Language records for Import Interval services and business rules **MUST** include the following:

- **Rule Type Code: Import Interval**

See **Create and Configure Rule Description Language** on page 6-12 for more information about creating RDLs and RDL Versions.

Configure RDLs

The next step is to configure each RDL Version that will be used. For Import Interval RDLs, this involves the following:

- Mapping data elements/attributes to tables/columns
- Configuring Table Validations
- Configuring Mappings
- Mapping interval data header information to the Header table
- Mapping interval values to the “T” table
- Configuring Interval Data Validations
- Validating RDL

See **RDL Configuration** on page 6-13 for more information about configuring RDLs.

Create Business Rule Record

The next step is to create the Business Rule records that represent the specific business rules.

Business Rule records for Import Interval services and business rules **MUST** include the following:

- **Rule Type: Import Interval**

See **Create and Configure Business Rule Records** on page 6-15 for more information about creating Business Rule records.

Define Import Interval Business Rule Properties and Parameters

As part of creating the Business Rule record, you must also define the Business Rule properties and parameters

Import Interval business rules use the following properties:

- **xmlstream:** Instructs the Adapter parsing engine to keep the XML data that is to be sent to the database engine
- **mode:** Instructs the Adapter parsing engine to perform database operations on a per table basis (“event”) or as a batch done at the end of the incoming payload document (“batch”).

Example:

```
<PROPERTIES>
  <PROPERTY NAME='xmlstream' VALUE='y' />
  <PROPERTY NAME='mode' VALUE='event' />
</PROPERTIES>
```

See **Defining Properties and Parameters for Adapter Business Rules** on page 11-114 and **Import/Interval Properties and Parameters** on page 11-116 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about business rule properties and parameters.

Associate Business Rule to RDL

The next step is to associate each Business Rule record to the corresponding RDL. See **Associating Business Rules to Rule Description Language** on page 11-66 in the *Oracle Utilities Energy Information Platform User's Guide* for more information.

Optional: Create XSLT Map Names, Map Versions, and Maps

If the business rule and service will use XSLT maps for transforming data, you must also create XSLT Map Name records, XSLT Map Versions, and XSLT Maps. See **Create and Configure XSLT Maps and Records** on page 6-17 for more information.

Create Runtime Service

The next step is to create a Runtime Service record for each business rule. See **Create and Configure Runtime Services** on page 6-18 for more information.

Define Service Properties

As part of creating the Runtime Service record, you must also define the Service properties. See **Service Properties** on page 6-19 and **Defining Service Properties for Adapter Runtime Services** on page 11-31 in the *Oracle Utilities Energy Information Platform User's Guide* for more information.

Create Service Activation record

The last step is to create a Service Activation record for each Runtime Service. See **Create Service Activation Records** on page 6-20 for more information.

Implementing Interval Services and Business Rules

The steps outlined below should be performed for **each** Import Interval business rule you wish to run using the Adapter.

Create RDLs and RDL Versions (optional)

The first step in implementing Import Interval services and business rules is to create and configure Rule Description Language (RDL) records for your Import Interval business rules.

Note: RDLs are optional for Import Interval business rules, and are needed only if performing Interval Data Validations.

Rule Description Language records for Import Interval services and business rules **MUST** include the following:

- **Rule Type Code: Interval**

See **Create and Configure Rule Description Language** on page 6-12 for more information about creating RDLs and RDL Versions.

Configure RDL

The next step is to configure each RDL Version that will be used. For Import Interval RDLs, this involves the following:

- Configuring Interval Data Validations
- Validating RDL

See **RDL Configuration** on page 6-13 for more information about configuring RDLs.

Create Business Rule Record

The next step is to create the Business Rule records that represent the specific business rules.

Business Rule records for Import Interval services and business rules **MUST** include the following:

- **Rule Type: Interval**

See **Create and Configure Business Rule Records** on page 6-15 for more information about creating Business Rule records.

Define Import Interval Business Rule Properties and Parameters

If importing interval data in LSE format, as part of creating the Business Rule record you must also define the appropriate Business Rule properties.

Import Interval business rules can use the following properties:

- **intdheadertable:** Specifies that interval data header information should be inserted into the table specified by this property instead of into the Channel Cut Header (LSCHANNELCUTHEADER) table. This property is used when importing data into either standard or enhanced interval data tables.
- **intdatatable:** Specifies that interval data information should be inserted into the table specified by this property instead of into the Channel Cut Data (LSCHANNELCUTDATA) table. **Note:** This property is NOT used when importing data into enhanced interval data tables.
- **filetype:** Instructs the Adapter that the interval data payload is in the Oracle Utilities Enhanced Input/Output (LSE) format.
- **XslConverter:** Optional property specifying the full class name of an XML converter to apply to the payload. See **XML Converters** on page 6-31 \ for more information about XML Converters.

Example:

```
<PROPERTIES>
  <PROPERTY NAME='filetype' VALUE='LSE' />
</PROPERTIES>
```

See **Defining Properties and Parameters for Adapter Business Rules** on page 11-114 and **Interval Properties and Parameters** on page 11-117 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about business rule properties and parameters.

Associate Business Rule to RDL (if applicable)

The next step is to associate each Business Rule record to the corresponding RDL. See **Associating Business Rules to Rule Description Language** on page 11-66 in the *Oracle Utilities Energy Information Platform User's Guide* for more information.

Note: This step is only necessary if using RDLs with Import Interval business rules.

Create Runtime Service

The next step is to create a Runtime Service record for each business rule. See **Create and Configure Runtime Services** on page 6-18 for more information.

Define Service Properties

As part of creating the Runtime Service record, you must also define the Service properties. See **Service Properties** on page 6-19 and **Defining Service Properties for Adapter Runtime Services** on page 11-31 in the *Oracle Utilities Energy Information Platform User's Guide* for more information.

Create Service Activation record

The last step is to create a Service Activation record for each Runtime Service. See **Create Service Activation Records** on page 6-20 for more information.

Implementing COM Services and Business Rules

The steps outlined below should be performed for **each** COM business rule you wish to run using the Adapter.

Create COM object

The first step is to create the COM object that the Adapter business rule will call. If you plan to use an existing COM object, you would create a COM wrapper interface for the Adapter to work with.

COM Object Requirements

The COM method to be invoked by the Adapter must contain a parameter list that matches the parameters the Adapter sends when executing a COM business rule.

The following is a C++ example of a valid parameter list:

```
HRESULT <METHOD>([in]IDispatch* pDisp,
                 [in]VARIANT xmlPayload,
                 [in]BSTR dataIn,
                 [in] BSTR optParam,
                 [out] BSTR *outBound,
                 [out,retval] BSTR *finished);
```

Parameter	Description
[in] IDispatch *	References an ADO database connection.

Parameter	Description
[in] VARIANT	Refers to the inbound payload. Can be either a IStream pointer or of string type.
[in] BSTR dataIn	This parameter is passed from the client process to the Adapter.
[in] BSTR optParam	This is data or parameters from the Optional Parameters field of the Business Rule record. This data is read from database. See COM Properties and Parameters on page 11-119 in the <i>Oracle Utilities Energy Information Platform User's Guide</i> for more information.
[out] BSTR *outbound	This is the outbound payload. The user must configure the “xmlstream” property to ‘y’ in order to use the outbound payload (see Define COM Business Rule Properties and Parameters on page 6-50 for more information). This variable is returned in the parameter list.
[out,retval] BSTR *finished	This XML contains the completion status of the COM method and any messages to describe a FAIL state. This variable is the return value from the COM method.

The first parameter list above ([in] IDispatch) is the ADO database connection object. In Visual Basic this would not be a dispatch pointer but an Object. Because this is not a new connection there is the potential to see any update activity on the database rolled back because of a FAIL status returned to the client process.

The second parameter is the inbound. This parameter is a VARIANT type because the Adapter supports a _IStream interface.

The next parameter is an input parameter that could contain data coming from the client process. It does not have to be in any particular format. This data is passed in “as is”.

The optParam parameters comes from the Optional Parameters field on the Business Record table in the database. The COM method has this data available for whatever use needed by the COM method.

The outbound parameter contains any BSTR data that needs to be reported back to the client process. This is not the return XML used to indicate the status of the COM call. It is not required that a COM method return an outbound payload. It is only necessary for there to be an outbound payload.

The last parameter contains the XML used to determine the completion status of the COM method call. This XML is required to be returned from the COM method. A DTD and example of the format is shown below. If the return XML is not present, then an exception is generated. If the status is FAIL, then the COM method should provide details within the Message element.

Completion Status XML - DTD

```
<!DOCTYPE BUSINESSRULE
[
<ELEMENT BUSINESSRULE(STATUS,MESSAGE)>
<ELEMENT STATUS(#PCDATA)>
<ELEMENT MESSAGE(#PCDATA)>
]>
```

Completion Status XML - Example

```
<BUSINESSRULE>
  <STATUS>PASS</STATUS>
  <MESSAGE/>
</BUSINESSRULE>
```

The STATUS element indicates either PASS or FAIL, and the MESSAGE should contain any error message that would detail a FAIL completion status to the client process.

Create Business Rule record

The next step is to create the Business Rule records that represent the specific business rules.

Business Rule records for COM services and business rules **MUST** include the following:

- **Rule Type: COM**

See **Create and Configure Business Rule Records** on page 6-15 for more information about creating Business Rule records.

Define COM Business Rule Properties and Parameters

As part of creating the Business Rule record, you must also define the Business Rule properties and parameters

COM business rules use the following properties:

- **progid:** The progid used by the Adapter to locate the COM component itself. The Adapter cannot run the COM method if it cannot find the COM component. If either the progid or the method name cannot be found, then an error is reported.
- **method:** A valid COM method with the appropriate number of parameters and the correctly formatted return XML.
- **xmlstream:** Used to return the completion status of the COM method call. If the user would like to be able to pass data back to a file or a message queue, then the outbound data and the xmlstream property are important.

Example:

```
<PROPERTIES>
  <PROPERTY NAME='progid' VALUE='echoTool.echoBox' />
  <PROPERTY NAME='method' VALUE='dbAction' />
  <PROPERTY NAME='xmlstream' VALUE='y' />
</PROPERTIES>
```

See **Defining Properties and Parameters for Adapter Business Rules** on page 11-114 and **COM Properties and Parameters** on page 11-119 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about business rule properties and parameters.

Create Runtime Service

The next step is to create a Runtime Service record for each business rule. See **Create and Configure Runtime Services** on page 6-18 for more information.

Define Service Properties

As part of creating the Runtime Service record, you must also define the Service properties. See **Service Properties** on page 6-19 and **Defining Service Properties for Adapter Runtime Services** on page 11-31 in the *Oracle Utilities Energy Information Platform User's Guide* for more information.

Create Service Activation record

The last step is to create a Service Activation record for each Runtime Service. See **Create Service Activation Records** on page 6-20 for more information.

Implementing Assembly Services and Business Rules

The steps outlined below should be performed for **each** Assembly business rule you wish to run using the Adapter.

Create .NET assembly

The first step is to create a .NET assembly that the Adapter business rule will call. The business rule can be built in any .Net language. The only requirements are the class must inherit the IBusinessRule interface, and it must have an Execute method with the parameters below:

- **Inputs:**
 - **Connection:** The data source and session information that can be accessed in the business rule. It includes the following information:
 - Data source name, password, qualifier, userid, ID, and connect string
 - Session
 - **Stream:** The payload that is getting passed into the rule. If it is getting called via an Adapter Service, it is the payload that is picked up from the file directory, queue, or database and the result of any input converters
 - **Configuration:** The Properties for the business rule. The values for this are set in the Properties field on the Business Rule.
 - **String:** The Parameters setup for the business rule. The values for this are set in the Optional Parameter field on the Business Rule.
- **Output String:** The output must include the response for the rule indicating whether it passed or failed and any associated messages. It can also include a payload that can be returned to the calling component.

The first step in building the rule is to determine the logic that needs to be in the rule along with the inputs and output, and any parameters and properties needed. The next step is to build the Assembly and define it in the Adapter. The final step is to setup the Adapter Service or Web Service that will call the Business Rule.

The following section will walk through the steps to build a sample Assembly Business rule.

Sample Assembly Rule

The sample rule will open, update or close a command using the Oracle Utilities Meter Data Management Command Tracking Interface.

The Connection input will be used to get the connection information needed to call the Command Tracking Interface.

The Stream input will include the payload and all the information required to log the command. It will also include an element indicating whether to open, update, or close the command. Below is sample payload to be used with this rule:

```
<SAMPLEREQUEST>
<WQITEM>
  <ACTION>OPEN</ACTION> - Indicates whether to OPEN, CLOSE, or UPDATE the
command
  <METERID VALTYPE="NEW">KRISTEN</METERID> - Meter ID use for the command
  <CHANNELID VALTYPE="NEW">1</CHANNELID> - Channel Id used for the command
  <UOMCODE VALTYPE="NEW">01</UOMCODE> - Unit of Measure
  <TYPE VALTYPE="NEW">METERPING</TYPE> - Type of command
  <QUEUE VALTYPE="NEW">MDM</QUEUE> - Queue for the command
  <UIDPHYSICALMETER VALTYPE="NEW">692</UIDPHYSICALMETER> - Physical Meter UID
  <NOTE>NEW Command</NOTE> - Note to be included with the command
</WQITEM>
</SAMPLEREQUEST>
```

The Configuration input will hold the web user and password required for the rule. The user ID and password will be stored as Properties in the business rule.

```
<PROPERTIES>
  <PROPERTY NAME="method" VALUE="Execute"/>
  <PROPERTY NAME="class" VALUE="MyCompany.BusinessRules.SampleAssemblyRule"/>
  <PROPERTY NAME="assembly" VALUE="SampleAssemblyRule, Version=1.0.0.0,
Culture=neutral, PublicKeyToken=e8aabe15cd1277a6"/>
  <PROPERTY NAME="SESSION_USER" VALUE="kristen"/>
  <PROPERTY NAME="SESSION_PASSWORD" VALUE="password99"/>
</PROPERTIES>
```

The String input value will indicate whether to automatically fail the rule. If the value is PASS then the rule will try and open, update, or close the command using the Command Tracking Interface. If it is FAIL then the rule will automatically through an error. The Optional Parameter field can be used to set this value.

The Output string is in the format of an XML response (MDMRESPONSE). It includes the STATUS (PASS or FAIL) along with the any errors or messages that need to be returned. If there is a failure, the error messages will be logged in the Inbound Exception (HUBINBOUNDEXCEPTION) table. Below is sample output string returned by this rule:

```
<MDMRESPONSE STATUS="FAIL">
  <ERRORS>
  <ERROR CODE="EXCPT" DESC="Error - Message" DATE="01-01-2000"> other text</
ERROR>
  </ERRORS>
</MDMRESPONSE>
```

The following is the sample assembly, which uses the values from the payload, properties, and optional parameters fields. Based on the ACTION parameter in the payload, it will open, close, or update a command via the Command Tracking Interface. Any exceptions are captured and returned to the calling component. If the rule is called via an Adapter Service, the exceptions will be logged to the Inbound Exception (HUBINBOUNDEXCEPTION) table and the Inbound (HUBINBOUD) record will have a status of "E". Refer to comments for more details.

```
using System.IO;
using System.Xml;
using System.Runtime;
using Lodestar.Adapter;
using Lodestar.Adapter.BusinessRules; // The Lodestar.Adapter.dll
must be referenced
using Lodestar.MDM; // Command Tracking Interface - the
Lodestar.MDM.dll must be referenced

using System;

// Names the Assembly and Class - Refer to the Properties field in
Figure 2. Business Rule Add/Edit - elements Class and Assembly
namespace MySampleCompany.BusinessRules
{
    public class SampleAssemblyRule : IBusinessRule<Stream, string>
    {
        public SampleAssemblyRule()
        {
        }

        // The Execute method is the method called - Refer to the
Properties field in Figure 2. Business Rule Add/Edit - element Method
        public string Execute(Connection cn, Stream payload,
Configuration configuration, string parameters)
        {
            string output = "";
            string response = "";
```

```

        string resp = "";
        string user = "";
        string password = "";

        try
        {
            // Checks the value in the paramters field for a value
            // of PASS or FAIL - Refer to the Optional Paramters field in Figure 2.
            // Business Rule Add/Edit
            if (parameters == "PASS")
            {
                //Load the payload that is passed into the business
                // rule from the Adpater Service or Web Service call
                XmlDocument myXMLDoc = new System.Xml.XmlDocument();
                MemoryStream memStream = (MemoryStream)payload;
                memStream.Seek(0, SeekOrigin.Begin);
                myXMLDoc.Load(memStream);
                XmlElement root = myXMLDoc.DocumentElement;

                // Gets the fields from the payload that are needed
                // to log the command via the Command Tracking Interface (CTI)
                XmlNode nNode1 =
                myXMLDoc.SelectSingleNode("SAMPLEREQUEST/WQITEM/METERID");
                string sMeter = nNode1.InnerText;
                XmlNode nNode2 =
                myXMLDoc.SelectSingleNode("SAMPLEREQUEST/WQITEM/CHANNELID");
                string sChannel = nNode2.InnerText;
                XmlNode nNode3 =
                myXMLDoc.SelectSingleNode("SAMPLEREQUEST/WQITEM/UOMCODE");
                string sUOM = nNode3.InnerText;
                XmlNode nNode4 =
                myXMLDoc.SelectSingleNode("SAMPLEREQUEST/WQITEM/TYPE");
                string sType = nNode4.InnerText;
                XmlNode nNode5 =
                myXMLDoc.SelectSingleNode("SAMPLEREQUEST/WQITEM/QUEUE");
                string sQueue = nNode5.InnerText;
                XmlNode nNode6 =
                myXMLDoc.SelectSingleNode("SAMPLEREQUEST/WQITEM/UIDPHYSICALMETER");
                string sPM = nNode6.InnerText;
                XmlNode nNode7 =
                myXMLDoc.SelectSingleNode("SAMPLEREQUEST/WQITEM/ACTION");
                string sCommand = nNode7.InnerText;
                XmlNode nNode8 =
                myXMLDoc.SelectSingleNode("SAMPLEREQUEST/WQITEM/NOTE");
                string sNote = nNode8.InnerText;

                //For the sample rule, the user name and password
                // are passed into the rule via the Properties field defined for the
                // business rule
                user = configuration.Property[3].Value;
                password = configuration.Property[4].Value;

                // Sets the connection string using the values
                // passed in via the connection parameter and properties
                string myConnection = "<DATASOURCE><NAME>" +
                cn.DataSource.Name + "</NAME>" +
                "<CONNECTSTRING>" +
                cn.DataSource.ConnectionString + "</CONNECTSTRING>" +
                "<QUALIFIER>" + cn.DataSource.Qualifier +
                "</QUALIFIER>" +
                "<USERID>" + user + "</USERID><PASSWORD>" +
                password + "</PASSWORD>" +

```

```

        "</DATASOURCE>";

        //Loads the CTI
        Lodestar.MDM.MDMTracking mc = new MDMTracking();
        // Using the connection defined above a session is
opened
        mc.OpenSession(myConnection);

        // For the Sample Rule the payload indicates whether
to open, update or cloase a command
        // Sets the value of the command paramters
        if (sCommand == "OPEN")
        {
            //Using the values passed in via the payload,
create the payload required to open a command
            string myCTIOpen = "<WQITEM><METERID>
VALTYPE=\"NEW\">" + sMeter + "</METERID>" +
                "<CHANNELID VALTYPE=\"NEW\">" + sChannel +
"</CHANNELID>" +
                "<UOMCODE VALTYPE=\"NEW\">" + sUOM + "</
UOMCODE>" +
                "<TYPE VALTYPE=\"NEW\">" + sType + "</TYPE>"
+
                "<QUEUE VALTYPE=\"NEW\">" + sQueue + "</
QUEUE>" +
                "<UIDPHYSICALMETER VALTYPE=\"NEW\">" + sPM
+ "</UIDPHYSICALMETER>" +
                "<NOTE VALTYPE=\"NEW\">" + sNote + "</NOTE>"
+
                "</WQITEM>";

            // Calls the CTI Open method
            output = mc.Open(myCTIOpen, true);
        }
        else if (sCommand == "UPDATE")
        {
            //Using the values passed in via the payload,
create the payload required to update a command
            string myCTIUpdate = "<WQITEM><METERID>" +
sMeter + "</METERID>" +
                "<CHANNELID>" + sChannel + "</CHANNELID>" +
                "<UOMCODE>" + sUOM + "</UOMCODE>" +
                "<TYPE VALTYPE=\"NEW\">" +
"METERPINGEXCEPTION" + "</TYPE>" +
                "<QUEUE>" + sQueue + "</QUEUE>" +
                "<UIDPHYSICALMETER>" + sPM + "</
UIDPHYSICALMETER>" +
                "<NOTE VALTYPE=\"NEW\">" + sNote + "</NOTE>"
+
                "</WQITEM>";

            // Calls the CTI Update method
            output = mc.Update(myCTIUpdate, true);
        }
        else if (sCommand == "CLOSE")
        {
            //Using the values passed in via the payload,
create the payload required to close a command
            string myCTIClose = "<WQITEM><METERID>" + sMeter
+ "</METERID>" +
                "<CHANNELID>" + sChannel + "</CHANNELID>" +
                "<UOMCODE>" + sUOM + "</UOMCODE>" +

```



```

        "<TYPE VALTYPE=\"NEW\">" +
"METERPINGEXCEPTION" + "</TYPE>" +
        "<QUEUE>" + sQueue + "</QUEUE>" +
        "<UIDPHYSICALMETER>" + sPM + "</
UIDPHYSICALMETER>" +
        "<NOTE VALTYPE=\"NEW\">" + sNote + "</NOTE>"
+
        "<USERID VALTYPE=\"NEW\">Test_User</USERID>"
+
        "</WQITEM>";

        // Calls the CTI Close method
        output = mc.Close(myCTIClose, true);
    }
    else
    {
        // If the command type passed in the payload is
not Open, Update, or Close the response will return an exception
        resp = "<OUTPUT><STATUS>FAIL</
STATUS><ERRORS><ERROR>Invalid Command</ERROR></ERRORS></OUTPUT>";
    }
    mc.CloseCurrentSession();

    // Checks the response of the CTI method to see if
it passed or failed. If it is OK then
    // the response will return a pass otherwise the
response will return an exception
    // Create Return String
    if (output == "OK")
    {
        resp = "<OUTPUT><STATUS>PASS</STATUS></OUTPUT>";
    }
    else
    {
        XmlDocument myXMLResponse = new
System.Xml.XmlDocument();
        myXMLResponse.LoadXml(output);
        XmlNode nNodeReturn =
myXMLResponse.SelectSingleNode("RETSTATUS");
        // Captures any critical exception and pass it
back to the calling component.
        // The HUBINBOUND record will show a status of
"E" and the errors messages will be logged in HUBINBOUNDEXCEPTION
        resp = "<MDMRESPONSE
STATUS=\"FAIL\"><ERRORS><ERROR CODE=\"EXCPT\" DESC=\"Error - " +
nNodeReturn.InnerText + "\" DATE=\"01-01-2000\">other text</ERROR></
ERRORS></MDMRESPONSE>";
    }
    }
    else
    {
        // Captures any critical exception and pass it back
to the calling component.
        // The HUBINBOUND record will show a status of "E"
and the errors messages will be logged in HUBINBOUNDEXCEPTION
        resp = "<MDMRESPONSE STATUS=\"FAIL\"><ERRORS><ERROR
CODE=\"EXCPT\" DESC=\"Parameter - " + parameters + "\" DATE=\"01-01-
2000\">other text</ERROR></ERRORS></MDMRESPONSE>";
    }
}

```

```
    }
    catch (Exception e)
    {
        // Captures any critical exceptions and passes them back
        to the calling component.
        // The HUBINBOUND record will show a status of "E" and
        the errors messages will be logged in HUBINBOUNDEXCEPTION
        resp = "<MDMRESPONSE STATUS=\"FAIL\"><ERRORS><ERROR
        CODE=\"EXCPT\" DESC=\"Error - \" + e.Message + \"\" DATE=\"01-01-
        2000\">other text</ERROR></ERRORS></MDMRESPONSE>";
    }
    // returns the response to the calling component
    return resp;
}
}
```

Install the Assembly

Once the Assembly is built, it needs to be put into the GAC. The GAC is located in the %WINDIR%\Assembly directory. Before an assembly can be placed into the GAC, the assembly must be signed. Signing the assembly provides the PublicKeyToken attribute to the full name and is done through the assembly properties menu.

Note: Each time a new assembly is installed the Adapter Service and ISS need to be restarted to see the updated version.

Another critical component is to ensure the Business Rule has the assembly name along with the Version, Culture and PublicKeyToken in the Business Rule Assembly property. The details of the full name can be found by looking at the properties of the assembly in the GAC.

```
<PROPERTIES>
  <PROPERTY NAME="method" VALUE="Execute"/>
  <PROPERTY NAME="class" VALUE="MyCompany.BusinessRules.SampleAssemblyRule"/>
  <PROPERTY NAME="assembly" VALUE="SampleAssemblyRule, Version=1.0.0.0,
  Culture=neutral, PublicKeyToken=e8aabe15cd1277a6"/>
  <PROPERTY NAME="SESSION_USER" VALUE="kristen"/>
  <PROPERTY NAME="SESSION_PASSWORD" VALUE="password99"/>
</PROPERTIES>
```

Create Business Rule record

The next step is to create the Business Rule records that represent the specific business rules.

Business Rule records for Assembly services and business rules MUST include the following:

- **Rule Type: Assembly**

Business Rule records for Assembly services and business rules can also include the following:

- **Optional Parameter:** Used to pass parameters to the rule

See **Create and Configure Business Rule Records** on page 6-15 for more information about creating Business Rule records.

Define Assembly Business Rule Properties and Parameters

As part of creating the Business Rule record, you must also define the Business Rule properties and parameters

Assembly business rules use the following properties:

- **assembly:** The .NET assembly
- **class:** The .NET class name
- **method:** The method to execute

- **WQUEUE:** Work Queue queue code for function calls and errors
- **WQTYPE:** Work Queue type used to identify function calls
- **WQEXCEPTIONTYPE:** Work Queue type used to identify errors

Example:

```
<PROPERTIES>
  <PROPERTY NAME='assembly' VALUE='Lodestar.MDM' />
  <PROPERTY NAME='class' VALUE='Lodestar.MDM.BusinessRules.MeterPingTWACS' />
  <PROPERTY NAME='method' VALUE='Execute' />
  <PROPERTY NAME='WQUEUE' VALUE='TWACS' />
  <PROPERTY NAME='WQTYPE' VALUE='METERPING' />
  <PROPERTY NAME='WQEXCEPTIONTYPE' VALUE='METERPINGEXCEPTION' />
</PROPERTIES>
```

In addition to these properties, assembly rules can include other properties to contain configuration input parameters required by the rule.

See **Defining Properties and Parameters for Adapter Business Rules** on page 11-114 and **Assembly Properties and Parameters** on page 11-118 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about business rule properties and parameters.

Create Runtime Service

The next step is to create a Runtime Service record for each business rule. See **Create and Configure Runtime Services** on page 6-18 for more information.

Define Service Properties

As part of creating the Runtime Service record, you must also define the Service properties. See **Service Properties** on page 6-19 and **Defining Service Properties for Adapter Runtime Services** on page 11-31 in the *Oracle Utilities Energy Information Platform User's Guide* for more information.

Create Service Activation record

The last step is to create a Service Activation record for each Runtime Service. See **Create Service Activation Records** on page 6-20 for more information.

Implementing CustomQuery Services and Business Rules

The steps outlined below should be performed for **each** CustomQuery business rule you wish to run using the Adapter.

Create custom query

The stored procedure executed by a CustomQuery business rule is defined in a file named **Procedures.*.xsl** in the C:\LODESTAR\CFG directory on the Adapter server, where the asterisk (*) is a user-defined name for the file.

In addition, the SQL statement executed by some business rules can be overridden by a user defined SQL statement. Allowing these queries to be modified enables the rules to function in environments where the customer has user defined meta-data.

To create a new custom stored procedure, locate one of the installed **Procedures.*.xsl** files in the C:\LODESTAR\Bin directory and copy it to the C:\LODESTAR\CFG directory, and then rename the file. You can edit existing stored procedures, or add new stored procedures.

To edit an existing stored procedure, find the stored procedure in the file and edit the SQL contained within the <sql> node. If adding a new stored procedure, insert a new <stored-procedure> node within the <stored-procedures> node. See the following example template:

```
<stored-procedure id="sp-YourProcedureName">
```

```
<xsl:stylesheet version="1.0" xmlns:spbc="urn:lsapps.spbccontext" exclude-
result-prefixes="spbc">
  <xsl:output omit-xml-declaration="yes"/>
  <xsl:template match="node()|/|@*">
    <query name="ResultNodeName">
      <sql>
        SELECT
        FROM
        WHERE
      </sql>
    </query>
  </xsl:template>
</xsl:stylesheet>
</stored-procedure>
```

Add the appropriate SQL statements within the `<sql>` node.

All parameters passed into the stored procedure are attributes of the template match. For example, to use the Serial Number parameter use the following syntax in the stored procedure:

```
<xsl:value-of select="@SERIALNUMBER"/>
```

Create Business Rule record

The next step is to create the Business Rule records that represent the specific business rules.

Business Rule records for CustomQuery services and business rules **MUST** include the following:

- **Rule Type: CustomQuery**

See **Create and Configure Business Rule Records** on page 6-15 for more information about creating Business Rule records.

Define CustomQuery Business Rule Properties and Parameters

As part of creating the Business Rule record, you must also define the Business Rule properties and parameters

Properties

CustomQuery business rules use the following properties:

- **STORED_PROCEDURE_NAME:** The name of the custom query to call

Example:

```
<PROPERTIES>
  <PROPERTY NAME='STORED_PROCEDURE_NAME' VALUE='sp-get-meters' />
</PROPERTIES>
```

Parameters

Parameters for CustomQuery business rules are based on the parameters used by the stored procedure method to be called.

Parameters for a custom query can be passed in through the payload or as part of the Optional Parameter field in the business rule configuration. The parameters must be in the form of "NAME" and "VALUE" pairs of "ATTRIBUTE" elements like below.

```
<ATTRIBUTES>
  <ATTRIBUTE NAME="ACCOUNTID" VALUE="9485763-2"/>
  <ATTRIBUTE NAME="CUSTOMERID" VALUE="102853AEG"/>
  <ATTRIBUTE NAME="METERID" VALUE="000452-432"/>
</ATTRIBUTES>
```

See **Defining Properties and Parameters for Adapter Business Rules** on page 11-114 and **CustomQuery Properties and Parameters** on page 11-120 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about business rule properties and parameters.

Create Runtime Service

The next step is to create a Runtime Service record for each business rule. See **Create and Configure Runtime Services** on page 6-18 for more information.

Define Service Properties

As part of creating the Runtime Service record, you must also define the Service properties. See **Service Properties** on page 6-19 and **Defining Service Properties for Adapter Runtime Services** on page 11-31 in the *Oracle Utilities Energy Information Platform User's Guide* for more information.

Create Service Activation record

The last step is to create a Service Activation record for each Runtime Service. See **Create Service Activation Records** on page 6-20 for more information.

Implementing Rate Services and Business Rules

The steps outlined below should be performed for **each** Rate business rule you wish to run using the Adapter.

Create and configure Rate Schedule

The first step when implementing a Rate business rule is to create and configure the rate schedule that will be executed by the Adapter.

Rate Schedule Requirements

To interface a rate schedule with the Adapter, the rate schedule must create a return XML string at the end of the rate schedule execution. This return XML string must indicate the completion status as either PASS or FAIL, any applicable error messages, and an optional return payload. Returning a payload to the Adapter allows the client process to handle data coming from the Rate Schedule. The outbound payload does not have to be in any particular format.

Using the Rules Language, the below set of instructions is one way to create the required XML.

```
BUSINESSRULE.STATUS = "PASS";
BUSINESSRULE.MESSAGE = "";
BUSINESSRULE.PAYLOAD = "<OPTIONAL></OPTIONAL>";
SAVE BUSINESSRULE TO XML;
```

It's also possible to create the required XML using the XML Rules Language statements and functions. See **Appendix B: XML Statements and Functions** in the *Oracle Utilities Rules Language Reference Guide* for more information about using these functions.

The example below is what actually is returned from the Rules Language after processing the instructions above.

```
<RSID_VALUES>
  <RSID_STEM NAME="BUSINESSRULE">
    <RSID_VALUE NAME="STATUS" TYPE="S" VALUE="PASS"/>
    <RSID_VALUE NAME="MESSAGE" TYPE="S" VALUE=""/>
    <RSID_VALUE NAME="PAYLOAD" TYPE="S" VALUE="&lt;OPTIONAL&gt;&lt;/OPTIONAL&gt;"/>
  </RSID_STEM>
</RSID_VALUES>
```

The STATUS element is the completion status of the Rate Schedule. The MESSAGE property provides a space for the Rate Schedule to place any error messages needed by the client process to describe a FAIL condition. Also, the PAYLOAD item provides a means for the Rate Schedule to return data back into the client process if applicable.

Specifying the Rate Schedule to Process

When the Adapter processes a Rate business rule, the specific Rate Schedule it will execute must be defined in an XML string and included as either parameters in the Business Rule record, or as part of the incoming payload.

- **Business Rule Parameters:** To specify the rate schedule in the Business Rule record, you must include an XML string in the Optional Parameters field that defines the data source, rate schedule, and any arguments to be passed to the rate schedule. See **Define Rate Business Rule Properties and Parameters** on page 6-60 for more information about defining the rate schedule as part of the Business Rule record.
- **Incoming Payload:** To specify the rate schedule as part of the incoming payload, the payload itself must contain an XML string that defines the data source, rate schedule, and any arguments to be passed to the rate schedule. This option allows the rate schedule to be selected dynamically, based on the payload.

Create Business Rule Record

The next step is to create the Business Rule records that represent the specific business rules.

Business Rule records for Rate services and business rules **MUST** include the following:

- **Rule Type: Rate**

See **Create and Configure Business Rule Records** on page 6-15 for more information about creating Business Rule records.

Note: Business Rules of type “Rate” should not be used with web services.

Define Rate Business Rule Properties and Parameters

As part of creating the Business Rule record, you must also define the Business Rule properties and parameters.

Properties

Rate business rules use the following properties:

- **xmlstream:** Used to return an outbound payload to the client process. It is not mandatory to return a outbound payload.

Example:

```
<PROPERTIES>
  <PROPERTY NAME='xmlstream' VALUE='y' />
</PROPERTIES>
```

Parameters

Parameters for Rate business rules include an XML structure that defines the rate schedule to be run when the business rule is executed.

Example:

```
<LODESTAR_RUNSCHEDULE>
  <DATASOURCE>
    <NAME>Local</NAME>
    <CONNECTSTRING>Data Source=<data_source>;User
ID=<user_id>;Password=<password>;LSProvider=ODP;</CONNECTSTRING>
    <QUALIFIER>testq</QUALIFIER>
  </DATASOURCE>
  <TIMEOUT>3600</TIMEOUT>
  <CONFIG_FILE_NAME>c:\lodestar\cfg\lodestar.cfg</CONFIG_FILE_NAME>
  <SCHEDULE_ARGUMENTS>
    <INTERVAL_DATABASE>RDB</INTERVAL_DATABASE>
    <RATE_FORM_IDENTIFIER>GLOBAL:GLOBAL:TEST_SCHEDULE</RATE_FORM_IDENTIFIER>
    <START_DATE>2003/11/25</START_DATE>
    <SAVE_RESULTS/>
  </SCHEDULE_ARGUMENTS>
</LODESTAR_RUNSCHEDULE>
```

```
<INTERVAL_DATA_ERROR>3</INTERVAL_DATA_ERROR>
<ACCOUNT><ALL/></ACCOUNT>
</SCHEDULE_ARGUMENTS>
</LODESTAR_RUNSCHEDULE>
```

Note that these parameters do not need to be included in the Business Rule record, but can instead be provided as part of the payload being processed by the Adapter.

See **Defining Properties and Parameters for Adapter Business Rules** on page 11-114 and **Rate Properties and Parameters** on page 11-122 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about business rule properties and parameters.

Create Runtime Service

The next step is to create a Runtime Service record for each business rule. See **Create and Configure Runtime Services** on page 6-18 for more information.

Define Service Properties

As part of creating the Runtime Service record, you must also define the Service properties. See **Service Properties** on page 6-19 and **Defining Service Properties for Adapter Runtime Services** on page 11-31 in the *Oracle Utilities Energy Information Platform User's Guide* for more information.

Create Service Activation record

The last step is to create a Service Activation record for each Runtime Service. See **Create Service Activation Records** on page 6-20 for more information.

Implementing Oracle Utilities Meter Data Management Services and Business Rules

These are business rules used by the Oracle Utilities Meter Data Management application.

Meter Data Management business rules are used with Oracle Utilities Meter Data Management to import usage data and perform other tasks. See **Chapter 5: Setting Up Oracle Utilities Meter Data Management Processes** in the *Oracle Utilities Meter Data Management Installation and Configuration Guide* for information about implementing Meter Data Management Services and Business Rules.

See **Chapter 7: Setting Up Oracle Utilities Meter Data Management Web Services** in the *Oracle Utilities Meter Data Management Installation and Configuration Guide* for information about implementing Oracle Utilities Meter Data Management web services and business rules.

Business Rule Examples

This section provides examples that illustrate how to set up records for RDLs (and related RDL Configurations), Business Rules, and Runtime Services for each type of business rule supported by the Adapter.

Refer to **Chapter 11: Setting Up and Configuring the Energy Information Platform Adapter** in the *Oracle Utilities Energy Information Platform User's Guide* for more information about creating the records described below.

Import Business Rule Example

Business Requirement: Import meter read data in comma delimited format from a file system.

About the Payload Data

In this example, the incoming delimited payload will be converted to XML using the ColumnCountCSVParser. See **CSV Parsers** on page 6-22 for more information.

Incoming Data:

```
BXT01_RES1,METERSRUS,12345,54321,07/30/2004 00:00:00,1,1500,07/2004
...
```

This data contains the following values (separated by commas):

```
<Meter ID>,<Manufacturer>,<Serial Number>,<UNI Number>,<Read Time>,<Bill
Determinant Code>,<Stop Reading>,<Read Month>
```

Converted Data:

```
<csv>
  <row>
    <col_0>BXT01_RES1</col_0> (Meter ID)
    <col_1>METERSRUS</col_1> (Manufacturer)
    <col_2>12345</col_2> (Serial Number)
    <col_3>54321</col_3> (UNI Number)
    <col_4>07/30/2004 00:00:00</col_4> (Read Time)
    <col_5>1</col_5> (Bill Determinant Code)
    <col_6>1500</col_6> (Stop Reading)
    <col_7>07/2004</col_7> (Read Month)
  </row>
</csv>
```

Rule Description Language Record

Create a Rule Description Language record as follows:

- **Rule Name:** Import_Meter_Read_CSV
- **Rule Type Code:** Import

RDL Version Record

Create an RDL Version record as follows:

- **RDL Name:** Import_Meter_Read_CSV
- **Start Time:** 11/01/2004
- **Stop Time:** N/A
- **Note:** Version 1

RDL Configuration

Configure the RDL as follows:

- Use the XML structure above (Converted Data) as the Template Source.
- Select the METERREAD table as the Template Destination.
- Create Table and Column mappings as follows:

Source (elements)	Destination	Action or Mapping Type
ROW	METERREAD (table)	Insert or Update
COL_0	METERID (column)	Direct
COL_1	MANUFACTURER (column)	Direct
COL_2	SERIALNO (column)	Direct
COL_3	UNINUMBER (column)	Direct
COL_4	METERREADTIME (column)	Direct
COL_5	BILLDETMCODE (column)	Direct
COL_6	STOPREADING (column)	Direct
COL_7	METERREADMONTH (column)	Direct

- Create a Table Validation on the METERREAD table as follows:
 - **Name:** Meter ID
 - **Group Parent:** Validations
 - **Member of Group:** Meter ID Validations
 - **Type:** SQL
 - **Error Code:** 100
 - **Error Description:** Meter ID %%METERREAD.METERID%% does not exist.
 - **Enabled:** Yes (checked)
 - **SQL Statement:**

```
select meterid from meter where meterid = '%%METERREAD.METERID%%'
```

Business Rule Record

Create a Business Rule record as follows:

- **Class Name:** N/A
- **Rule Name:** Import_Meter_Read_CSV
- **Description:** Import Meter Read Data - CSV
- **Rule Type:** Import
- **Properties:**

```
<PROPERTIES>
  <PROPERTY NAME="xmlstream" VALUE="y"/>
  <PROPERTY NAME="mode" VALUE="BATCH"/>
</PROPERTIES>
```

Business Rule to RDL Association Record

Create a Business Rule to RDL Association record as follows:

- **Business Rule:** Import_Meter_Read_CSV
- **RDL Name:** Import_Meter_Read_CSV

Runtime Service Record

Create a Runtime Service record as follows:

- **Name:** Import_Meter_Read_CSV
- **JMS Queues:** N/A
- **JMS Filter:** N/A
- **JMS Parameters:** -server -Djava.endorsed.dirs=./lib
- **Origination System:** FILELOADER
- **Parameters:** N/A
- **Internal Flag:** Y
- **Business Rule Flag:** N
- **Output Flag:** N
- **Runtime Service Type:** FPORTAL
- **Properties:**

```
<PROPERTIES>
  <PROPERTY NAME="STORAGE_LEVEL" VALUE="2"/>
  <PROPERTY NAME="RULE_NAME" VALUE="Import_Meter_Read_CSV"/>
  <PROPERTY NAME="RDL_MEMORY_SIZE" VALUE="3"/>
  <PROPERTY NAME="INPUT_DC_0"
VALUE="com.lodestarcorp.core.xml.parsers.ColumnCountCSVParser"/>
  <PROPERTY NAME="POLL_DIRECTORY" VALUE="C:\Import\CSV"/>
  <PROPERTY NAME="POLL_EXTENSION" VALUE="txt"/>
  <PROPERTY NAME="POLL_INTERVAL" VALUE="10"/>
</PROPERTIES>
```

Service Activation Record

Create a Service Activation record as follows:

- **Server:** <SERVER_NAME>
- **Runtime Services:** Import_Meter_Read_CSV
- **Enabled:** Yes
- **Run at Startup:** Yes
- **Keep Alive:** Yes

Import Interval Business Rule Example

Business Requirement: Import interval data in XML format from a file system.

About the Payload Data

In this example, the incoming payload data is already in XML format. Interval data to be imported should always have an SPI value of 1800 (30 minute data).

Incoming Data:

```
<INTERVAL_DATA>
<CUT>
  <RECORDER>BXT01_RES1</RECORDER>
  <CHANNEL>1</CHANNEL>
  <STARTTIME>2001-03-01T00:00:00.000</STARTTIME>
  <STOPTIME>2001-03-31T23:59:59.000</STOPTIME>
  <DST_PARTICIPANT>N</DST_PARTICIPANT>
  <VALIDATION_REQUIRED>N</VALIDATION_REQUIRED>
  <PULSE_MULTIPLIER>1</PULSE_MULTIPLIER>
  <PULSE_OFFSET>0</PULSE_OFFSET>
  <SPI>1800</SPI>
  <UOM>1</UOM>
  <TIMEZONE>0</TIMEZONE>
  <TIME_ZONE_STANDARD_NAME />
  <TIMESTAMP>2001-12-06T10:37:50.000</TIMESTAMP>
  <ORIGIN>C</ORIGIN>
  <DESCRIPTOR>'UMS INC'</DESCRIPTOR>
  <INTERVAL>
    <RECORDING>
      <VALUE>16308</VALUE>
    </RECORDING>
  </INTERVAL>
  ...
</CUT>
<INTERVAL_DATA>
```

Rule Description Language Record

Create a Rule Description Language record as follows:

- **Rule Name:** Import_INTD_XML
- **Rule Type Code:** Import Interval

RDL Version Record

Create an RDL Version record as follows:

- **RDL Name:** Import_INTD_XML
- **Start Time:** 11/01/2004
- **Stop Time:** N/A
- **Note:** Version 1

RDL Configuration

Configure the RDL as follows:

- Use the XML structure above (Converted Data) as the Template Source. The Template Destination are the Header and I tables, provided by default.
- Create Table and Column mappings as follows:

Source (elements)	Destination	Action or Mapping Type
CUT	HEADER (table)	Insert Cut
RECORDER	RECORDER (column)	Direct
CHANNEL	CHANNEL (column)	Direct
STARTTIME	STARTTIME (column)	Direct
STOPTIME	STOPTIME (column)	Direct
DST_PARTICIPANT	DST (column)	Direct
SPI	SPI (column)	Direct
UOM	UOM (column)	Direct
TIME_ZONE_STANDARD_NAME	TZSN (column)	Direct
RECORDING	I (table)	Insert Read
VALUE	V (column)	Direct

- Create an Interval Data Validation as follows:
 - **Settings:** Validate SPI
 - **Seconds:** 1800

Business Rule Record

Create a Business Rule record as follows:

- **Class Name:** N/A
- **Rule Name:** Import_INTD_XML
- **Description:** Import Interval Data - XML
- **Rule Type:** Import Interval
- **Properties:**

```
<PROPERTIES>
  <PROPERTY NAME="xmlstream" VALUE="y"/>
  <PROPERTY NAME="mode" VALUE="BATCH"/>
</PROPERTIES>
```

Business Rule to RDL Association Record

Create a Business Rule to RDL Association record as follows:

- **Business Rule:** Import_INTD_XML
- **RDL Name:** Import_INTD_XML

Runtime Service Record

Create a Runtime Service record as follows:

- **Name:** Import_INTD_XML
- **JMS Queues:** N/A
- **JMS Filter:** N/A
- **JMS Parameters:** -server -Djava.endorsed.dirs=./lib
- **Origination System:** FILELOADER
- **Parameters:** N/A
- **Internal Flag:** Y
- **Business Rule Flag:** N
- **Output Flag:** N
- **Runtime Service Type:** FPORTAL
- **Properties:**

```
<PROPERTIES>
  <PROPERTY NAME="STORAGE_LEVEL" VALUE="2"/>
  <PROPERTY NAME="RULE_NAME" VALUE="Import_INTD_XML"/>
  <PROPERTY NAME="RDL_MEMORY_SIZE" VALUE="3"/>
  <PROPERTY NAME="POLL_DIRECTORY" VALUE="C:\Import\INTD"/>
  <PROPERTY NAME="POLL_EXTENSION" VALUE="XML"/>
  <PROPERTY NAME="POLL_INTERVAL" VALUE="10"/>
</PROPERTIES>
```

Service Activation Record

Create a Service Activation record as follows:

- **Server:** <SERVER_NAME>
- **Runtime Services:** Import_INTD_XML
- **Enabled:** Yes
- **Run at Startup:** Yes
- **Keep Alive:** Yes

Interval Business Rule Example

Business Requirement: Import interval data in LSE format from the file system. Exceptions generated from this business rule should be sent to the Work Queues.

About the Payload Data

In this example, the incoming payload data is delivered in the LSE format. The interval data to be imported should always have an SPI value of 3600 (1 hour data).

Rule Description Language Record

Create a Rule Description Language record as follows:

- **Rule Name:** Import_INTD
- **Rule Type Code:** Interval

RDL Version Record

Create an RDL Version record as follows:

- **RDL Name:** Import_INTD
- **Start Time:** 11/01/2004
- **Stop Time:** N/A
- **Note:** Version 1

RDL Configuration

Configure the RDL as follows:

- Create an Interval Data Validation as follows:
 - **Settings:** Validate SPI
 - **Seconds:** 3600

Business Rule Record

Create a Business Rule record as follows:

- **Class Name:** N/A
- **Rule Name:** Import_INTD
- **Description:** Import Interval Data - LSINTD
- **Rule Type:** Interval
- **Properties:**

```
<PROPERTIES>
  <PROPERTY NAME='filetype' VALUE='LSE' />
</PROPERTIES>
```

Business Rule to RDL Association Record

Create a Business Rule to RDL Association record as follows:

- **Business Rule:** Import_INTD
- **RDL Name:** Import_INTD

Runtime Service Record

Create a Runtime Service record as follows:

- **Name:** Import_INTD
- **JMS Queues:** N/A
- **JMS Filter:** N/A
- **JMS Parameters:** -server -Djava.endorsed.dirs=./lib
- **Origination System:** FILELOADER
- **Parameters:** -N/A
- **Internal Flag:** Y
- **Business Rule Flag:** N
- **Output Flag:** N
- **Runtime Service Type:** FPORTAL
- **Properties:**

```
<PROPERTIES>
  <PROPERTY NAME="STORAGE_LEVEL" VALUE="2"/>
  <PROPERTY NAME="RULE_NAME" VALUE="Import_INTD"/>
  <PROPERTY NAME="RDL_MEMORY_SIZE" VALUE="3"/>
  <PROPERTY NAME="POLL_DIRECTORY" VALUE="C:\Import\LSE"/>
  <PROPERTY NAME="POLL_EXTENSION" VALUE="LSE"/>
  <PROPERTY NAME="POLL_INTERVAL" VALUE="10"/>
  <PROPERTY NAME="WQ_TYPE" VALUE="Invalid_SPI"/>
  <PROPERTY NAME="WQ_QUEUE" VALUE="INTD_Validations"/>
</PROPERTIES>
```

Service Activation Record

Create a Service Activation record as follows:

- **Server:** <SERVER_NAME>
- **Runtime Services:** Import_INTD
- **Enabled:** Yes
- **Run at Startup:** Yes
- **Keep Alive:** Yes

COM Business Rule Example

Business Requirement: Validate usage data sent to the Ice Payload table using a custom COM component.

About the Payload Data

In this example, the incoming payload data is sent to the Ice Payload table in the Oracle Utilities Data Repository.

Business Rule Record

Create a Business Rule record as follows:

- **Class Name:** N/A
- **Rule Name:** VAL_Usage
- **Description:** Validate Usage Data
- **Rule Type:** COM
- **Properties:**

```
<PROPERTIES>
  <PROPERTY NAME='progid' VALUE='XSVAL.VALDATA' />
  <PROPERTY NAME='method' VALUE='Validate' />
  <PROPERTY NAME='xmlstream' VALUE='y' />
</PROPERTIES>
```

Runtime Service Record

Create a Runtime Service record as follows:

- **Name:** VAL_Usage
- **JMS Queues:** N/A
- **JMS Filter:** N/A
- **JMS Parameters:** -server -Djava.endorsed.dirs=./lib
- **Origination System:** DATALOADER
- **Parameters:**
- **Internal Flag:** Y
- **Business Rule Flag:** N
- **Output Flag:** N
- **Runtime Service Type:** DPORTAL
- **Properties:**

```
<PROPERTIES>
  <PROPERTY NAME="STORAGE_LEVEL" VALUE="2" />
  <PROPERTY NAME="RULE_NAME" VALUE="VAL_Usage" />
  <PROPERTY NAME="RDL_MEMORY_SIZE" VALUE="3" />
  <PROPERTY NAME="PAYLOAD_FORMAT" VALUE="B" />
  <PROPERTY NAME="PAYLOAD_TYPE" VALUE="DATAVAL" />
  <PROPERTY NAME="POLL_INTERVAL" VALUE="60" />
</PROPERTIES>
```


Service Activation Record

Create a Service Activation record as follows:

- **Server:** <SERVER_NAME>
- **Runtime Services:** VAL_Usage
- **Enabled:** Yes
- **Run at Startup:** Yes
- **Keep Alive:** Yes

Rate Business Rule Example

Business Requirement: Validate interval data in the Oracle Utilities Data Repository using a rate schedule (VAL_INTD) upon successful import. This process is invoked by a message in a JMS queue (QVALINTD) inserted after import of data by another Adapter business rule.

About the Payload Data

In this example, the incoming payload data (which specifies the interval data cuts to validate) is sent to a JMS queue (QVALINTD).

Business Rule Record

Create a Business Rule record as follows:

- **Class Name:** N/A
- **Rule Name:** Validate_INTD
- **Description:** Validate Interval Data
- **Rule Type:** Rate
- **Properties:**

```
<PROPERTIES>
<PROPERTY NAME='xmlstream' VALUE='y' />
</PROPERTIES>
```
- **Parameters:**

```
<LODESTAR_RUNSCHEDULE>
  <DATASOURCE>
    <NAME>Local</NAME>
    <CONNECTSTRING>Data Source=<data_source>;User
ID=<user_id>;Password=<password>;LSProvider=ODP;</CONNECTSTRING>
    <QUALIFIER>testq</QUALIFIER>
  </DATASOURCE>
  <TIMEOUT>3600</TIMEOUT>
  <CONFIG_FILE_NAME>c:\lodestar\cfg\lodestar.cfg</CONFIG_FILE_NAME>
  <SCHEDULE_ARGUMENTS>
    <INTERVAL_DATABASE>RDB</INTERVAL_DATABASE>
    <RATE_FORM_IDENTIFIER>GLOBAL:GLOBAL:VAL_INTD</RATE_FORM_IDENTIFIER>
    <START_DATE>2003/11/25</START_DATE>
    <SAVE_RESULTS/>
    <INTERVAL_DATA_ERROR>3</INTERVAL_DATA_ERROR>
  </SCHEDULE_ARGUMENTS>
</LODESTAR_RUNSCHEDULE>
```

Runtime Service Record

Create a Runtime Service record as follows:

- **Name:** Validate_INTD
- **JMS Queues:** QVALINTD
- **JMS Filter:** N/A
- **JMS Parameters:** -server -Djava.endorsed.dirs=./lib
- **Origination System:** QLOADER
- **Parameters:** -qs QVALINTDOUT
- **Internal Flag:** Y
- **Business Rule Flag:** N
- **Output Flag:** N
- **Runtime Service Type:** QPORTAL
- **Properties:**

```
<PROPERTIES>
  <PROPERTY NAME="STORAGE_LEVEL" VALUE="2"/>
  <PROPERTY NAME="RULE_NAME" VALUE="Import_INTD"/>
  <PROPERTY NAME="RDL_MEMORY_SIZE" VALUE="3"/>
</PROPERTIES>
```

Service Activation Record

Create a Service Activation record as follows:

- **Server:** <SERVER_NAME>
- **Runtime Services:** Import_INTD
- **Enabled:** Yes
- **Run at Startup:** Yes
- **Keep Alive:** Yes

Assembly Business Rule Example

Business Requirement: Validate usage data sent to the Ice Payload table using a custom .NET assembly.

About the Payload Data

In this example, the incoming payload data is sent to the Ice Payload table in the Oracle Utilities Data Repository.

Business Rule Record

Create a Business Rule record as follows:

- **Class Name:** N/A
- **Rule Name:** VAL_Usage
- **Description:** Validate Usage Data
- **Rule Type:** Assembly

- **Properties:**

```
<PROPERTIES>
  <PROPERTY NAME='assembly' VALUE='Assembly.VAL' />
  <PROPERTY NAME='class' VALUE='Assembly.VAL.BusinessRules.VALIDATE' />
  <PROPERTY NAME='method' VALUE='Validate' />
  <PROPERTY NAME='WQQUEUE' VALUE='USAGE_VAL' />
  <PROPERTY NAME='WQTYPE' VALUE='USAGE_VAL' />
  <PROPERTY NAME='WQEXCEPTIONTYPE' VALUE='USAGE_VAL_EXCEPTION' />
</PROPERTIES>
```

Runtime Service Record

Create a Runtime Service record as follows:

- **Name:** VAL_Usage
- **JMS Queues:** N/A
- **JMS Filter:** N/A
- **JMS Parameters:** -server -Djava.endorsed.dirs=./lib
- **Origination System:** DATALOADER
- **Parameters:**
- **Internal Flag:** Y
- **Business Rule Flag:** N
- **Output Flag:** N
- **Runtime Service Type:** DPORTAL
- **Properties:**

```
<PROPERTIES>
  <PROPERTY NAME="STORAGE_LEVEL" VALUE="2" />
  <PROPERTY NAME="RULE_NAME" VALUE="VAL_Usage" />
  <PROPERTY NAME="RDL_MEMORY_SIZE" VALUE="3" />
  <PROPERTY NAME="PAYLOAD_FORMAT" VALUE="B" />
  <PROPERTY NAME="PAYLOAD_TYPE" VALUE="DATAVAL" />
  <PROPERTY NAME="POLL_INTERVAL" VALUE="60" />
</PROPERTIES>
```

Service Activation Record

Create a Service Activation record as follows:

- **Server:** <SERVER_NAME>
- **Runtime Services:** VAL_Usage
- **Enabled:** Yes
- **Run at Startup:** Yes
- **Keep Alive:** Yes

CustomQuery Business Rule Example

Business Requirement: Execute a custom query to return records from the Oracle Utilities Data Repository.

About the Payload Data

In this example, the incoming payload data is sent to the Ice Payload table in the Oracle Utilities Data Repository.

Business Rule Record

Create a Business Rule record as follows:

- **Class Name:** N/A
- **Rule Name:** Get_Meter_Data
- **Description:** Get Usage Data
- **Rule Type:** CustomQuery
- **Properties:**

```
<PROPERTIES>
  <PROPERTY NAME='STORED_PROCEDURE_NAME' VALUE='sp-get-meters' />
</PROPERTIES>
```

Runtime Service Record

Create a Runtime Service record as follows:

- **Name:** Get_Meter_Data
- **JMS Queues:** N/A
- **JMS Filter:** N/A
- **JMS Parameters:** -server -Djava.endorsed.dirs=./lib
- **Origination System:** DATALOADER
- **Parameters:**
- **Internal Flag:** Y
- **Business Rule Flag:** N
- **Output Flag:** N
- **Runtime Service Type:** DPORTAL
- **Properties:**

```
<PROPERTIES>
  <PROPERTY NAME="STORAGE_LEVEL" VALUE="2"/>
  <PROPERTY NAME="RULE_NAME" VALUE="Get_Meter_Data"/>
  <PROPERTY NAME="RDL_MEMORY_SIZE" VALUE="3"/>
  <PROPERTY NAME="PAYLOAD_FORMAT" VALUE="B"/>
  <PROPERTY NAME="PAYLOAD_TYPE" VALUE="DATAVAL"/>
  <PROPERTY NAME="POLL_INTERVAL" VALUE="60"/>
</PROPERTIES>
```

Service Activation Record

Create a Service Activation record as follows:

- **Server:** <SERVER_NAME>
- **Runtime Services:** Get_Meter_Data
- **Enabled:** Yes
- **Run at Startup:** Yes
- **Keep Alive:** Yes

The Adapter Server

The Adapter Server is the core runtime engine that runs the Adapter. The Adapter Server must be running on each application server on which you wish to run the Adapter.

Set Up the LTMH.CFG.XML File

The LTMH.CFG.XML file is a configuration file used to specify the database connection used by the Adapter. The LTMH.CFG.XML file must be installed in the **C:\LODESTAR\CFG** directory on each application server on which the Adapter will run.

Set up this file on each application server you plan to use as an Adapter server.

LTMH.CFG.XML Example

```
<LTMH JMSEENABLED="false">
  <!-- Uncomment and customize lines appropriate to your installation -->
  <!-- Connection for an Oracle database -->
  <JDBC URL="jdbc:oracle:thin:@localhost:1521:orcl">
    <DRIVER TYPE="ORACLE">oracle.jdbc.OracleDriver</DRIVER>
    <QUALIFIER>tr450</QUALIFIER>
    <USERNAME>tr450</USERNAME>
    <PASSWORD>password</PASSWORD>
  </JDBC>
  <!-- Optional database connection information
    - Service property settings will override these
    - Uncomment the appropriate section(s) -->
  <DEFAULT>
    <ORACLECONNECTION>
      <DSN>local</DSN>
      <QUALIFIER>tr450</QUALIFIER>
      <USERNAME>tr450</USERNAME>
      <PASSWORD>password</PASSWORD>
    </ORACLECONNECTION>
  </DEFAULT>
</LTMH>
```

See **LTMH.CFG.XML** on page 2-46 for more information about setting up this file.

Starting the Adapter Server

To start the Adapter Server:

1. Select **Start->Programs->Oracle Utilities->LTMHServer**.
The **LTMH Server** window opens on the desktop.
2. The window displays commands used by each of the services currently configured to run on the server.
3. When a service is started, the Adapter Server loads the properties of the service, and starts the service.

Note that a service can only be started if it has been properly configured, and if all external components and/or systems are in place and configured. For instance, if a service is using the File Portal (FPORTAL) to poll a specified directory, the service will not start if that directory does not exist.

4. As the Adapter processes business rules, the Adapter Server window displays processing details, including results of applying an RDL (when applicable), any error messages generated, and other information.

The Adapter Monitor

The Adapter Monitor allows administrators to monitor Adapter runtime services and message queues. The Adapter Monitor consists of three tabs:

- **Runtime Services Tab**
- **Runtime Queue Tab**
- **Messaging Tool Tab**

Note that the Adapter Server must be running in order to start the Adapter Monitor.

Starting the Adapter Monitor

To start the Adapter Monitor:

1. Select **Start->Programs->Oracle Utilities->LTMHMon**.

The **Oracle Utilities Transaction Management Monitor** application opens.

2. Select the server you wish to monitor from the Adapter Server list and click **Connect**.

The Runtime Services tab lists the services configured to run on the selected server.

To disconnect from an Adapter Server, select the server and click **Disconnect**.

Runtime Services Tab

The Runtime Services tab allows you to monitor, start, stop, and refresh Adapter runtime services.

For each runtime service active on the Adapter Server, the Runtime Services tab displays the following information:

- **Service Name:** The name of the service.
- **Queue Name:** The message queue associated with the service (if applicable).
- **Status:** The current status of the service. Status is indicated by color:
 - **Green:** Running
 - **Red:** Stopped

How to start a service:

1. Select the service in the Service Name column and click **Start**. To start all services, click **Start All**.

How to stop a currently active service:

1. Select the service in the Service Name column and click **Stop**. To stop all currently running services, click **Stop All**.

How to refresh a currently active service:

1. Select the service in the Service Name column and click **Refresh**.

How to enable automatic refresh for all active services:

1. Select **Tools->Auto-Refresh Services**.

Runtime Queue Tab

The Runtime Queue tab allows you to monitor and refresh Adapter message queues. This tab is only available when using the Adapter with JMS queues.

For each runtime queue, the Runtime Queue tab displays the following information:

- **Queue Name:** The message queue.
- **Messages:** The number of messages in the message queue.

How to refresh a message queue:

1. Select the queue in the Queue Name column and click **Q Refresh**.

How to enable automatic refresh for all active queues:

1. Select **Tools->Auto-Refresh Queues**.

Messaging Tool Tab

The Messaging Tool tab allows you to work with messages in the Adapter message queues. This tab is only available when using the Adapter with JMS queues.

The Messaging Tool tab provides a number of functions for working with messages and message queues.

How to work with the Messaging Tool:

1. Select the message queue you wish to work with from the **JMS Queues** list. If sending a message, type the message in the message box below the JMS Queues list.
2. Click the appropriate function button.

Function	Description
Send	Sends a message to a specified message queue. The message is typed in the message box below the JMS Queues list.
Receive	Removes and destroys the first available message in the selected queue (used primarily for testing).
Browse	Displays all messages in the selected queue.
Flush	Destroys all messages in the selected queue.
Clear	Clears the message box below the JMS Queues list.
Load	Loads multiple messages in the selected queue.

Securing and Encrypting the Adapter Server and Monitor

This section describes how to secure and encrypt communication between the Adapter Server and Monitor using Secure Socket Layer (SSL), including:

- **Technical Overview**
- **Creating Certificates**
- **Enabling Secure Communications**

Technical Overview

Java Secure Socket Extension (JSSE) is used to implement SSL in the Adapter. JSSE is a set of packages that enable secure network communications and provide a framework and an implementation for a Java version of the SSL protocol, including functionality for data encryption, server authentication, message integrity and optional client authentication.

A "certificate" is a digitally signed statement vouching for the identity and public key of an entity. Certificates can either be self-signed or issued by a Certification Authority (CA).

- Certificates are stored in LTMH\Runtime\security\ltnh.jks (referred to as the "keystore").
- Trusted Certificate Authority (CA) certificates are stored in LTMH\Runtime\security\ltnh.cacerts (referred to as the "truststore").

Creating Certificates

Securing communications between the Adapter Server and Adapter Monitor requires creating a certificate that validates the identity and public key of the server. Certificates can be created using the **keytool.bat** batch file located in the ..\LODESTAR\LTMH\Runtime directory on the application server.

Use the following procedure to create a certificate on an application server

1. Open a command prompt on the application server, and navigate to the ..\LODESTAR\LTMH\Runtime directory.
2. Enter the following command:

```
keytool.bat -gensign -alias <cert_alias> -keystorepass lodestar  
-trustpass lodestar
```

where:

Parameter	Description
<cert_alias>	The alias for the certificate to be created. This is the value that will be entered in the Certificate Alias field on the Server record (see Enabling Secure Communications on page 6-81). Use a unique alias for each server.

3. Answer the questions as prompted by the program.
4. The program will generate a Certificate Signing Request (CSR) file in the ..\LODESTAR\LTMH\Runtime directory. The filename for this file is <cert_alias>.csr.
5. Send the content of CSR file to Certificate Authority (CA). The CA will send a signed certificate.
6. Import the signed certificate to the keystore using the **keytool.bat** program as follows:

```
keytool.bat -importsigned -alias <cert_alias> -keystorepass  
lodestar -file signedFromCSA.csr
```

where:

Parameter	Description
<cert_alias>	The alias for the certificate (from step 2 above).

- Successful operation should have the following messages:

Importing signed certificate

Certificate reply was installed in keystore

Securing Communications Between Multiple Servers

If you use multiple application servers in your environment, you must copy the certificates for each to the keystores on the other machines. Use the following procedure to copy certificates between application servers:

- Create a certificate on each server. Be sure to use a different (unique) alias on each server.
- Copy the `..\LODESTAR\LTMH\Runtime\security\lthm.jks` keystore file from one application server to the other.
- Import the certificate from the copied keystore to the local keystore using the **keytool.bat** batch file, as follows:

```
keytool.bat -importkeysignd -alias <cert_alias> -keystorepass
lodestar -srckeystore <copied_key>
```

where:

Parameter	Description
<cert_alias>	The alias for the certificate to be imported.
<copied_key>	The file location of copied keystore.

Set up LTMH.CFG.XML Configuration File

Once the certificates are in place (within the keystore and truststore files), the Adapter Server and Adapter Monitor must be able to communicate with the files. To enable this, you must define the password for each file in the LTMH.CFG.XML configuration file.

LTMH.CFG.XML Example

```
<LTMH JMSENABLED="false">
  <!-- Passwords for Keystore and TrustStore -->
  <SECURITY>
    <KEYSTORE><PASSWORD>lodestar</PASSWORD></KEYSTORE>
    <TRUST><PASSWORD>lodestar</PASSWORD></TRUST>
  </SECURITY>
  ...
</LTMH>
```

See **LTMH.CFG.XML** on page 2-46 for more information about setting up this file.

Changing Passwords for Keystore and Truststore

The default password for the keystore (the file that stores certificates) and truststore (the file that stores Trust Certificate Authority certificates) is “lodestar.” These passwords can be changed using the **keytool.bat** batch file as follows:

```
keytool.bat -cpkey <new_keypw> -cptrust <new_trpw>
```

where:

Parameter	Description
<new_keypw>	The new password for the keystore.
<new_trpw>	The new password for the truststore.

Enabling Secure Communications

You enable secure communications between the Adapter Server and Adapter Monitor by specifying the certificate used on the application server.

How to enable secure communications:

1. Select **Tools and Utilities->Adapter Components->Servers**.
2. Search for and select the server you wish to secure.
3. Enter the the certificate alias for the server in the **Certificate Alias** field.

You define a certificate alias using the keytool.bat file (located in the ..\LODESTAR\LTMH\Runtime directory on the application server).

See **Creating Certificates** on page 6-79 above for more information about creating certificates and certificate aliases.

4. Specify whether or not to allow unsigned certificates on the **Allow unsigned certificate** drop-down list.

If the certificate referred to by the certificate alias is unsigned and the value for **Allow unsigned certificate** field is Yes, then communication is allowed. However, a message on both Adapter Server and Adapter Monitor will appear telling that the certificate is unsigned. If the value is No, the Adapter Server will shutdown and no communication is allowed.

Chapter 7

Configuring Energy Information Platform Security

This chapter describes how to configure Security to work with the Energy Information Platform features, including:

- **Security Features**
- **Framework Features**
- **Entity Management Features**
- **Common Features**
 - **Adapter Features**
- **Work Queue Features**
 - **Securing Work Queue Items**
- **Adapter Features**
- **Data Source Features**
- **Securing Reports**

Important Notes about Assigning Features Privileges

By default, the feature permissions restrict access to all functions and screens. To allow users access to the Energy Information Platform functionality, create users and/or groups and enable appropriate features for each.

Security Features

Security features include:

- **Security:** Enables the Security Admin menu option.
 - **Data Sources:** Enables the Data Sources tab and functions.
 - **Delete:** Enables deleting of data sources.
 - **Groups:** Enables the Groups tab and functions.
 - **Delete:** Enables deleting of groups.
 - **Users:** Enables the Users tab and functions.
 - **Delete:** Enables deleting of users.
 - **Data Privileges:** Enables the Data Privileges screen and functions.
 - **Delete:** Enables deleting of Data Privileges.
 - **Sessions:** Enables the Sessions tab and functions.
 - **Delete:** Enables deleting of sessions.

Framework Features

Framework features include:

- **Banner Items:** Enables the icons on the Top Menu of the Energy Information Platform user interface.
 - **Home:** Displays and enables the Home icon.
 - **Help:** Displays and enables the Help icon.
 - **Set Default Menu:** Displays and enables the Set Default Menu icon (used to access Data Navigator).
 - **Go to Security Administration:** Displays and enables the Security Administration icon.
 - **Current Database:** Displays and enables the Current Database icon.
 - **Themes:** Displays and enables the Themes icon.
 - **Alert:** Displays and enables the Alert icon.
- **Hide Disabled Menu Items:** When this option is selected, disabled menu items do not appear on the Left Menu.
- **Clips:** Enables access to clips via the Catalog tab of the dashboard screen.
 - **Related Market Participants:** Displays and enables the Related Market Participants clip.
 - **Quote of the day:** Displays and enables the Quote of the day clip.
 - **Current Message:** Displays and enables the Current Message clip.
 - **Server Time:** Displays and enables the Server Time clip.
 - **Loading Statistics:** Displays and enables the Loading Statistics clip.
 - **Process Control Interface:** Displays and enables the Process Control Interface clip.
 - **User Info:** Displays and enables the User Info clip.
 - **Favorite Accounts:** Displays and enables the Favorite Accounts clip.
 - **Work Queue Type Code:** Displays and enables the Work Queue Type Code clip.
 - **Work Queues:** Displays and enables the Work Queues clip.
 - **WQ Stat Users:** Displays and enables the WQ Stat Users clip.
 - **Work Queues Unassigned:** Displays and enables the Work Queues Unassigned clip.
 - **Contract:** Displays and enables the Contract clip.
 - **Expired Contract:** Displays and enables the Expired Contract clip.
 - **WQ Approvals:** Displays and enables the WQ Approvals clip.
 - **Settlement:** Displays and enables the Settlement clip.
 - **MDM Process Statistics:** Displays and enables the MDM Process Statistics clip.

User Preferences Features

User Preferences features include Set Preferences and Options features.

Set Preferences Features

Set Preferences features include:

- **User Context:** Enables the User Context tab and function.
- **Locales:** Enables the Locales tab and function.
- **Reset Password:** Enables the Reset Password tab and function.

Options

The User Preference Options include:

- **General Options:** Enables the General Options menu item.
 - **Set Default User Options:** Enables the Set Default User Options function on the General Options screen.
- **Billing Options:** Enables the Billing Options menu item.
 - **Set Default User Options:** Enables the Set Default User Options function on the Billing Options screen.
- **Report Options:** Enables the Report Options menu item.

Entity Management Features

This section describes the securable features of the Entity Management functions of the Energy Information Platform.

Entity Management features include:

- **Customer:** Enables the Customers menu item.
 - **Search Customers:** Enables the Search, insert, update and deletion functions on the Customers screen.
 - **Insert:** Enables the Add Customer option on the Customer screen.
 - **Update:** Enables the Save option on the Customer screen.
 - **Delete:** Enables the Delete option on the Customer screen.
 - **Clips:** Enables the Customer dashboard and related clips
 - **Customer Basics:** Displays and enables the Customer Basics clip.
 - **Customer Accounts:** Displays and enables the Customer Accounts clip.
 - **Customer Contacts:** Displays and enables the Customer Contacts clip.
 - **Customer Messages:** Displays and enables the Customer Messages clip.
 - **Customer Contracts:** Displays and enables the Customer Contracts clip.
- **Accounts:** Enables the Accounts menu item.
 - **Search Accounts:** Enables the Search function on the Accounts screen.
 - **Insert:** Enables the Add option on the Account screen.
 - **Update:** Enables the Save option on the Account screen.
 - **Delete:** Enables the Delete option on the Account screen.
 - **Clips:** Enables the Account dashboard and related clips
 - **Account Basics:** Displays and enables the Account Basics clip.
 - **Meters:** Displays and enables the Meters clip.
 - **Recorders:** Displays and enables the Recorders clip.
 - **Contacts:** Displays and enables the Contacts clip.
 - **Related Service Point:** Displays and enables the Related Service Point clip.
 - **Account Reports:** Displays and enables the Account Reports clip.
 - **Related Accounts:** Displays and enables the Related Accounts clip.
 - **Account Messages:** Displays and enables the Account Messages clip.
 - **Account Billing:** Displays and enables the Account Billing clip.
 - **Account Contracts:** Displays and enables the Account Contracts clip.
 - **Financial:** Displays and enables the Financials clip.
 - **Aggregation:** Displays and enables the Aggregation clip.
 - **Meter Data Management:** Displays and enables the Meter Data Management clip.
- **Recorders:** Enables the Interval Data menu item (on the Usage menu).
 - **Search Recorders:** Enables the Search function on the Usage screen.

- **Analysis Graph:** Enables the Analysis Graph link on the Interval Data Results screen used to view graphs of selected cut series data.
- **Download CSV:** Enables the Download CSV link on the Interval Data Results screen used to create a *.CSV file of selected cut series data.
- **Meters:** Enables the Meters menu item.
 - **Search Meters:** Enables the Search function on the Meters screen.
- **Customer Component** permissions include:
 - **Basics:** Enables the Basics clip and screen on the Customer Summary widow.
 - **Contacts:** Displays contact information on the Contacts clip and screen of the Customer Summary window.
 - **Insert:** Enables the Add Existing and Create contact functions on the Contacts screen.
 - **Update:** Enables the modify contacts function on the Contacts screen.
 - **Remove:** Enables the Remove function on the Contacts screen.
 - **Accounts:** Enables the Accounts clip and screen on the Customer Summary window.
 - **Insert:** Enables the Add Account function on the Customers screen.
- **Account Component** permissions include:
 - **Basics:** Enables the Basics clip and screen on the Account Summary window.
 - **Contacts:** Displays contact information on the Contacts screen.
 - **Insert:** Enables the Add Existing and Create contact functions on the Contacts screen.
 - **Update:** Enables the modify contacts function on the Contacts screen.
 - **Remove:** Enables the Remove function on the Contacts screen.
 - **Meters:** Enables the Meters clip and screen on the Account Summary window.
 - **Insert:** Enables the Add function on the Meters screen.
 - **Update:** Enables the modify function on the Meters screen.
 - **Remove:** Enables the Delete function on the Meters screen.
 - **Recorders:** Enables the Basics and screen on the Account Summary window.
- **Market Participants** permissions include:
 - **Search Market Participant:** Enables the Market Participants option on the Entity Management menu.
 - **Insert:** Enables the Add option on the Market Participant screen.
 - **Update:** Enables the user to edit the Market Participant record.
 - **Delete:** Enables the user to delete the Market Participant record
 - **Clips:** Enables the Market Participant dashboard and related clips
 - **Market Participant Basics:** Enables the Contacts clip and screen on the Market Participant Summary window.
 - **Market Participant Contacts:** Also enables the Contacts clip and screen on the Market Participant Summary window.
 - **Related Market Participants:** Enables the Related Market Participants clip and screen on the Market Participant Summary window.

- **Service Points:** Enables the Service Points clip and screen on the Market Participant Summary window.
- **Meters:** Enables the Meters clip and screen on the Market Participant Summary window.
- **Recorders:** Enables the Recorders clip and screen on the Market Participant Summary window.
- **Transactions:** Enables the Transactions clip and screen on the Market Participant Summary window.
- **Service Points** permissions include:
 - **Search Service Point:** Enables the Service Points option on the Entity Management menu.
 - **Insert:** Enables the Add option on the Service Points screen.
 - **Update:** Enables the user to edit the Service Point record.
 - **Delete:** Enables the user to delete the Service Point record.
 - **Clips:** Enables the Service Point dashboard and related clips.
 - **Service Point Basics:** Enables the Service Point Basics clip and screen on the Service Point Summary window.
 - **Service Point Contacts:** Enables the Contacts clip and screen on the Service Point Summary window.
 - **Service Point Market Participants:** Enables the Market Participants clip and screen on the Service Point Summary window.
 - **Meters:** Enables the Meters clip and screen on the Service Point Summary window.
 - **Recorders:** Enables the Recorders clip and screen on the Service Point Summary window.
 - **Service Point Contracts:** Enables the Contracts clip and screen on the Service Point Summary window.
 - **Meter Data Management:** Displays and enables the Meter Data Management clip.
 - **Transactions:** Enables the Transactions clip and screen on the Service Point Summary window.

Common Features

This section describes the securable features of the Tools and Utilities functions of the Energy Information Platform, including:

- **List Management:** Enables the Lists menu item and function.
 - **Insert:** Enables the Add function on the Lists screen.
 - **Update:** Enables the edit function on the List screen.
 - **Delete:** Enables the Delete function on the List screen.
 - **Search List:** Enables the Search function on the Lists screen.
 - **Add Account:** Enables the Add Items function on the Lists screen for Account lists.
 - **Remove Account:** Enables the Remove Selected Items function on the Lists screen for Account lists.
 - **Add Customer:** Enables the Add Items function on the Lists screen for Customer lists.
 - **Remove Customer:** Enables the Remove Selected Items function on the Lists screen for Customer lists.
 - **Edit Query:** Enables users to edit the queries defined in query lists.
- **Address Book:** Enables the Address Book menu item on the Tools and Utilities menu.
 - **Search Contacts:** Enables the Search Contacts screen.
 - **Add:** Enables the Add option in the Contact Methods pane of the Search Contact screen.
 - **Delete:** Enables the Delete option in the Contact Methods pane of the Search Contact screen.
 - **Edit Contacts:** Enables the Edit Contact screen.
 - **Save:** Enables the Save option on the Edit Contact screen.
 - **Contact Methods:** Enables the Contact Methods pane on the Edit Contact screen.
 - **Add:** Enables the Add option in the Contact Methods pane of the Edit Contact screen.
 - **Delete:** Enables the Delete option in the Contact Methods pane of the Edit Contact screen.
 - **Address Information:** Enables the Address Information pane on the Edit Contact screen.
 - **Add:** Enables the Add option in the Address Information pane of the Edit Contact screen.
 - **Add Existing:** Enables the Add Existing option in the Address Information pane of the Edit Contact screen.
 - **Delete:** Enables the Delete option in the Address Information pane of the Edit Contact screen.
 - **Contact Groups:** Enables the Groups menu item.
 - **Add:** Enables the Add option on the Search Groups screen.
 - **Update:** Enables the Save option on the Edit Group screen.
 - **Delete:** Enables the Delete option on the Search Groups result screen.
 - **Add Item:** Allows users to add contacts to a contact group on the Edit Group screen.

- **Delete Item:** Allows users to delete contacts from contact groups on the Edit Group screen.
- **Email Aliases:** Allows users to create email aliases from contact groups on the Edit Group screen.
- **Configuration:** Enables the Configuration menu option on the Tools and Utilities menu.
 - **Season Schedule:** Enables the Season Schedule menu option and function.
 - **TOU Schedule:** Enables the TOU Schedule menu option and function.
 - **TOU Period:** Enables the TOU Period Name menu option and function.
- **Report Admin:** Enables the Report Admin menu option.
 - **Super User:** Allows user to view all reports (even those that are not shared).
- **Email:** Enables the Send Email option on the View Report screen.
- **Scheduler:** Enables the Schedule Report option on the Run Reports screen.
 - **Recurrence Pattern - None:** Enables the Recurrence Pattern - None option on the Schedule Process screen.
 - **Recurrence Pattern - Intra-Daily:** Enables the Recurrence Pattern - Intra-Daily option on the Schedule Process screen.
 - **Recurrence Pattern - Daily:** Enables the Recurrence Pattern - Daily option on the Schedule Process screen.
 - **Recurrence Pattern - Weekly:** Enables the Recurrence Pattern - Weekly option on the Schedule Process screen.
 - **Recurrence Pattern - Monthly:** Enables the Recurrence Pattern - Monthly option on the Schedule Process screen.
- **Run Reports:** Enables the Run Reports menu option on the Tools and Utilities menu.
- **View Reports:** Enables the View Reports menu option on the Tools and Utilities menu.
- **Usage:** Enables the Usage menu
 - **Center:** Enables the Center option on the Usage menu.
 - **Interval Usage:** Enables the Interval Usage menu option.
 - **Add Cuts:** Enables the Add option on the Usage screen.
 - **Search Recorders:** Enables the Search option on the Usage screen.
 - **Modify Cuts:** Enables the Edit Cut option on the Usage screen.
 - **Delete Cuts:** Enables the Delete option on the Usage screen.
- **Workset:** Enables the Workset menu item and related functions.
 - **Data Source Cuts:** Enables the Data Source Cuts menu on the Workset screen.
 - **Save:** Enables the Save option on the Data Source Cuts menu.
 - **Restore:** Enables the Restore Archive option on the Data Source Cuts menu.
 - **Delete:** Enables the Delete option on the Data Source Cuts menu.
 - **Create Keys List:** Enables the View Keys Lists function on the Data Source Cuts drop-down list.
 - **Workset Cuts:** Enables the Workset Cuts menu on the Workset screen.
 - **Upload File:** Enables the Upload File option on the Data Source Cuts menu on the Workset screen.

- **Search:** Enables the Search option on the Workset screen.
- **Pagination:** Enables the pagination option on the Workset screen.
- **Interval Data:** Enables the Interval Data option on the Usage menu
- **Scalar Data:** Enables the Scalar Data menu on the Usage menu
 - **Meters:** Enables the Meters option on the Scalar Data sub-menu.
 - **Meter Values:** Enables the Meter Values option on the Scalar Data sub-menu.
- **Import/Export:** Enables the Import/Export menu on the Usage menu
 - **Import Data:** Enables the Import Data option on the Import/Export sub-menu.
 - **Export Data:** Enables the Export Data option on the Import/Export sub-menu.
 - **Setup:** Enables the Setup menu on the Import/Export sub-menu.
 - **Export Entity:** Enables the Export Entity option on the Import/Export sub-menu.
 - **Export Purpose:** Enables the Export Purpose menu on the Import/Export sub-menu.
- **Upload Data:** Enables the Upload Data option on the Usage menu
- **Run Reports:** Enables the Run Reports menu option on the Usage menu.
- **View Reports:** Enables the View Reports menu option on the Usage menu.

Securing Interval Data Sources

By default, users with privileges to use the Interval Data Manager and Interval Data Viewer functions of the Energy Information Platform can access all interval data sources defined in the Interval Data Sources table. If necessary, access to these interval data sources can be restricted through use of record-level security (see **Working with Record-Level Security** on page 12-9 for more information about record-level security). For example, specific groups or users could be restricted to working with only a subset of interval data sources, while other groups or users might have access to a different subset of interval data sources.

Access to interval data sources can be secured by applying Direct security to the Interval Data Sources table. This involves securing the table when configuring Direct security for the data source, and granting Data Access Privileges to the table for the appropriate users and/or groups.

When working with a relatively small number of interval data sources (less than fifty), the simplest approach is to define specific records (Keys) that each group or user can access.

Alternatively, you could secure the table to allow access to only those records of a specified Type (Relational or BTE), or that have a certain Read Only or Audit flag. These would most likely take the form of a SQL statement that defines the allowed access to the table.

Work Queue Features

This section describes the securable features of the Work Queues functions of the Energy Information Platform and how to restrict access data to work queue items using record-level security.

Work Queues Features

Work Queues features include:

- **WorkQueues:** Enables the Work Queues menu option.
 - **My WQ Items:** Enables the My WQ Items menu item and function.
 - **My WQ Approvals:** Enables the My WQ Approvals menu item and function.
 - **Search WQ Items:** Enables the Search WQ Items menu item and function.
 - **Search WQ Approvals:** Enables the Search WQ Approvals menu item and function.
 - **Search WQ Close:** Enables the Search WQ Closed menu item and function.
 - **WQ Actions:** Enables the Work Queue actions on views and search results screens.
 - **Assign:** Enables the Assign action.
 - **Unassign:** Enables the Unassign action.
 - **Approve:** Enables the Approve action.
 - **Reject:** Enables the Reject action.
 - **Resolve:** Enables the Resolve action.
 - **Close:** Enables the Close action.
 - **Reopen:** Enables the Reopen action.
 - **Hint Link:** Enables hint links on work queue items.
 - **Modify:** Enables the Modify action.
 - **WQ Views:** Enables the My WQ Items and My WQ Approvals view screens and the search results Display options from the Search WQ Items, Search WQ Approvals, and Search WQ Closed options.
 - **Groups:** Enables the Groups tab.
 - **Summary:** Enables the Summary tab.
 - **Details:** Enables the Details tab.
 - **Common Product:** Enables the WQ Items and My WQ Items menu options on the product menus (Billing, Pricing and Contracts, Meter Data Management, etc.).
 - **Run Reports:** Enables the Run Reports menu option on the Work Queues menu.
 - **View Reports:** Enables the View Reports menu option on the Work Queues menu.

Securing Work Queue Items

By default, users with privileges to use the Work Queues functions of the Energy Information Platform can access all open work queue items (stored as records in the Work Queue Open Items table). If necessary, access to work queue items can be restricted through use of record-level security (see **Working with Record-Level Security** on page 12-9 for more information about record-level security). For example, access could be based on Work Queue Type that restrict specific users to view and work with only those work queue items of a specified type or types. Likewise, access could be based on both the Work Queue Type and Work Queue that restricts users to those work queue items of a specified Work Queue Type(s) and Work Queue(s).

Access to work queue items can be secured using either Direct or Indirect security.

Using Direct Security

Directly securing the Work Queue Open Items table involves securing the table when configuring Direct security for the data source, and granting Data Access Privileges to the table for the appropriate users and/or groups. For example, you might secure the table to allow access to only those records of a specified Work Queue Type or Work Queue. This will most likely take the form of a SQL statement that defines the allowed access to the table.

Keep in mind that using Direct security means that only the Work Queue Open Items table will be secured. This approach does not restrict access to records in the Work Queue Closed Items table, or to records in any parent/lookup tables. Securing those tables would require applying Direct security to those tables as well.

Using Indirect Security

Indirectly securing the Work Queue Open Items table involves securing one or more of its parent table(s) when configuring Direct security for the data source, and granting Data Access Privileges to the parent table(s) for the appropriate users and/or groups. For example, you might secure the Work Queue table to allow access to a specific Work Queue, and via Indirect security, allow access to only those Work Queue Open Item records of the specified Work Queue.

Parent/lookup tables for the Work Queue Open Items table include the following:

- Work Queue (see note below)
- Work Queue Type
- Product
- Business Process

Using Indirect security to secure the Work Queue Open Items table allows you to restrict access to work queue records (both open and closed) at a higher level than individual work queue items. For example, by applying direct security to the Work Queue Type table, you restrict access that table, as well as the Work Queue Open Items table and the Work Queue Closed Items table. One possible scenario for this approach is one where each user or group has its own work queue type (defined in the Work Queue Type table).

Note: Securing Via the Work Queue Table

The Work Queue table is a parent of both the Work Queue Open Items table and the Work Queue Type table (which itself is a parent of the Work Queue Open Items table). Securing the Work Queue table also secures both of these tables via inherited indirect security, and will restrict access to records in the Work Queue Open Items table by both Work Queue and Work Queue Type.

Adapter Features

This section describes the securable features of the Adapter of the Energy Information Platform. Adapter features include:

- **Configure Adapter Rules:** Enables the Configure Adapter Rules menu option.
 - **Business Rule:** Enables the Business Rule menu item and function.
 - **Rule Description Language:** Enables the Rule Description Language menu item and function.
 - **Configure RDL:** Enables the Configure RDL option.
 - **Update:** Enables updating and saving of RDL Configuration.
 - **Global Function:** Enables the Global Functions menu item and function.
 - **Commands:** Enables the Commands menu item and function.
 - **Configure:** Enables the Configure option on the Commands menu.
 - **Execute:** Enables executing of commands from the Commands menu.
- **LTMH/Adapter Components:** Enables the Oracle Utilities Transaction Management/Adapter Components menu option.
 - **Business Rules:** Enables the Business Rule menu item and function.
 - **Servers:** Enables the Servers menu item and function.
 - **JMS Servers:** Enables the JMS Servers menu item and function.
 - **JMS Queues:** Enables the JMS Queues menu item and function.
 - **Payload Types:** Enables the Payload Types menu item and function.
 - **Exception Types:** Enables the Exception Types menu item and function.
 - **Origination Systems:** Enables the Origination Systems menu item and function.
 - **System Properties:** Enables the System Properties menu item and function.
 - **Runtime Services:** Enables the Runtime Services menu item and function.
 - **Service Activation:** Enables the Service Activation menu item and function.
 - **XSLT Map Names:** Enables the XSLT Map Names menu item and function.
 - **XSLT Map Versions:** Enables the XSLT Map Versions menu item and function.
 - **XSLT Maps:** Enables the XSLT Map menu item and function.
 - **Trading Partners:** Enables the Trading Partners menu item and function.
 - **External Transaction Types:** Enables the External Transaction Types menu item and function.
 - **Web Services:** Enables the Web Services menu option
 - **Configuration:** Enables the Web Services menu option and function
 - **Execution:** Enables users/groups to execute web services
- **Adapter Runtime:** Enables the Adapter Runtime menu option.
 - **Search Inbound Records:** Enables the Search menu option.

Data Source Features

This section describes how to secure data source access in Data Navigator. Data Source permissions include:

- **Data Source:** Enables the Data button on the Left menu, and grants access to the Data menu used by Data Navigator. If one or more tables in the currently selected data source has been secured, the Data Source node appears as a link. Clicking the link opens the Navigator Features digraph, which illustrates dependencies between secured tables.
- **<TABLE>:** Enables Data Navigator access to the specified table, where <TABLE> is one of the tables within the currently selected data source.
 - **Insert:** Enables the Add function in the Data Navigator for records in the parent table.
 - **Update:** Enables the Save function in the Data Navigator for records in the parent table.
 - **Delete:** Enables the Delete function in the Data Navigator for records in the parent table.
- **<COLUMN>:** Enables Data Navigator access to the specified column, where <COLUMN> is one of the columns in a table within the currently selected data source.
 - **Search View:** Displays the column on Data Navigator search screens.
 - **List View:** Displays the column on Data Navigator list screens.
 - **Detail View:** Displays the column on Data Navigator edit screens.
 - **Editable:** Enables users to edit the data in the column. When this option is disabled, the column is effectively “Read-Only.”

Securing Reports

This section describes how security can be applied to reports created using the Report Framework, including Crystal Reports and Oracle Utilities Rules Language reports. This includes:

- **Overview**
- **Securing the Contents of the Run Reports Screen**
- **Securing the Contents of the View Reports Screen**

Overview

By default, users with privileges to use the Run Reports and View Reports functions available through the Energy Information Platform (including the Run Reports and View Reports options on product menus such as Pricing and Contracts, Billing, etc.) can run and view all available reports.

If necessary, access to running and viewing reports can be restricted through use of record-level security (see **Working with Record-Level Security** on page 12-9 for more information about record-level security). For example, access could be based on the Report Template that restricts specific users to run only those reports based on a specified template. Likewise, view access could be based on the type of report (Crystal, Active, LSRate, etc.), the report title (if specific naming conventions for report titles are used), or the user(s) who created the report(s).

Securing the Contents of the Run Reports Screen

Access to running reports via the Run Reports screen can be secured using either Direct or Indirect security.

Using Direct Security

Directly securing the contents of the Run Reports screen involves securing the Report Template (LSRFRPTTEMPLATE) table when configuring Direct security for the data source, and granting Data Access Privileges to the table for the appropriate users and/or groups. For example, you might secure the table to allow access to only those reports of a specified type (Crystal or LSRate) or reports with a specific naming convention. This will most likely take the form of a SQL statement that defines the allowed access to the table.

Keep in mind that using Direct security means that only the Report Template table will be secured. This approach does not restrict access to reports in the Report Instance table (via the View Reports screen), or to records in any parent/lookup tables. Securing those tables would require applying Direct security to those tables as well.

Using Indirect Security

Indirectly securing the Report Template table involves securing its one parent table, Report Type (LSRFRPTTYPE) when configuring Direct security for the data source, and granting Data Access Privileges to the parent table(s) for the appropriate users and/or groups.

For example, you would secure the Report Type table to allow access to a specific report type (Crystal, LSRate, etc.), and via Indirect security, allow access to only those report templates of the specified report type.

Securing the Contents of the View Reports Screen

Access to viewing reports via the View Reports screen can be secured using either Direct or Indirect security.

Using Direct Security

Directly securing the contents of the View Reports screen involves securing the Report Instance (LSRFRPTINSTANCE) table when configuring Direct security for the data source, and granting Data Access Privileges to the table for the appropriate users and/or groups. For example, you might secure the table to allow access to only those reports of a specified type (Crystal or LSRate) or reports with a specific naming convention (either in the Report Name or Report Title field). This will most likely take the form of a SQL statement that defines the allowed access to the table.

Keep in mind that using Direct security means that only the Report Instance table will be secured. This approach does not restrict access to reports in the Report Template table (via the Run Reports screen), or to records in any parent/lookup tables. Securing those tables would require applying Direct security to those tables as well.

Using Indirect Security

Indirectly securing the Report Instance table involves securing its one parent table, Report Type (LSRFRPTTYPE) when configuring Direct security for the data source, and granting Data Access Privileges to the parent table(s) for the appropriate users and/or groups.

For example, you would secure the Report Type table to allow access to a specific report type (Crystal, LSRate, etc.), and via Indirect security, allow access to only those report instances of the specified report type.

Chapter 8

Setting Up Processing to run in Batch Mode

This chapter includes instructions and information detailing the use of a number of command line programs that perform specific functions in batch mode.

The following types of batch executables are described:

- **Data Utilities:** Tools for importing/exporting customer and interval data, rate forms, and updating customer/account lists.
- **Billing:** Batch executable programs that allow for automatic billing, approval required billing, current/final billing, and bill corrections.
- **Miscellaneous:** Miscellaneous tools, including Run Rate Schedules (RUNRS), autoprofiler, and RCTORDBM.

Batch Executables Overview

This section includes brief descriptions of the individual batch executables described in this book and in other User's Guides, as noted.

Data Utilities

Data utilities include the following tools for importing/exporting customer and interval data, rate forms, and updating customer/account lists.

- **PLIMPORT:** Used to import (add/update/delete) data into the Oracle Utilities Data Repository. PLIMPORT is described on page 8-4.
- **INTDIMP:** Used to import interval data into the Interval Database or Data Repository. INTDIMP is described on page 8-9.
- **INTDIMPEX:** Used to import interval data into an enhanced interval data table. INTDIMPEX is described on page 8-26.
- **INTDEXP:** Used to export and/or delete interval data from the Interval or Data Repository. INTDEXP is described on page 8-35.
- **RFIMPEXP:** Used to import/export rate forms into and out of the RCDATA directory and/or the Oracle Utilities Data Repository. RFIMPEXP is described in **Appendix A: Setting Up Rate Form Records and Rate Codes** in the *Oracle Utilities Rules Language User's Guide*.
- **EXPCA:** Used to export and/or delete customer/account data from the Oracle Utilities Data Repository. EXPCA is described on page 8-29.
- **EXPTBL:** Used to export and/or delete data from one or more specified tables in the Oracle Utilities Data Repository. EXPTBL is described on page 8-32.
- **LSTRFRSH:** Used to refresh customer/account lists. LSTRFRSH is described in **Chapter 8: Working with Lists and Queries** in the *Data Manager User's Guide*.
- **LSTIMEXP:** Used to import/export lists into and out of the RCDATA directory and/or the Oracle Utilities Data Repository. LSTIMEXP is described in **Chapter 8: Working with Lists and Queries** in the *Data Manager User's Guide*.

Billing

These batch executable programs allow for automatic billing, approval required billing, current/final billing, bill corrections, and mass billing. The billing batch executables are described in **Chapter 8: Setting Up Billing to run in Batch Mode** in the *Oracle Utilities Billing Component Installation and Configuration Guide*, and include:

- **AUTOBILL:** Used to automatically process bills in batch mode.
- **APRVREQ:** Used to process approval required bills in batch mode.
- **CURFINAL:** Used to process current and/or final bills in batch mode.
- **BILLCORR:** Used to process bill corrections, including Cancel, Adjustments, Rebills, and Cancel/Rebill, in batch mode.
- **MASSBILL:** Used to process accounts that are eligible for mass billing.

Miscellaneous

Miscellaneous programs described in this manual include:

- **RUNRS:** Used to execute Oracle Utilities Rules Language rate schedules. RUNRS is described on page 8-41.
- **AUTOPROF:** Used to estimate a load profile for an account by “spreading” its metered kWh value over a time period according to a class profile or other template. AUTOPROF is described on page 8-48.
- **RCTORDBM:** Used to move data, including rate forms, queries, lists, and other data, from the RCDATA directory to tables in the Oracle Utilities Data Repository. RCTORDBM is described on page 8-53.

Importing Data

This section describes how data can be imported into the Oracle Utilities Data Repository in batch mode. This includes:

- **Importing Customer Data**
- **Importing Interval Data**
- **Importing Enhanced Interval Data**

Importing Customer Data

Customer data can be imported into the Oracle Utilities Data Repository in batch mode using the PLIMPORT.EXE command line program.

Prerequisites

The data to be loaded must be in the Oracle Utilities Import format. See **Appendix B: Oracle Utilities Data Repository Schema Import File Format** for more information.

Note: Placing all data fields in double quotes (“ ”) minimizes the likelihood of data errors due to special characters; for example, “OP-CO”, “LU/C”, “Large Utility & Company”.

Important Note

All users should avoid working with the Browser while the Import is taking place. Otherwise, the Browser and the Import utility may conflict and contend for system resources.

PLIMPORT Command Syntax

PLIMPORT uses the syntax shown below. Parameter switches are case insensitive (you can enter them in either upper or lower case (-c or -C)).

```
plimport -f filename -d connectstring [-q qualifier] -m mode -t TABLENAME  
[-i configfilename] [-l logfilename] [-e errorfile] [-x maxerrors] [-b batchsize]  
[-a logrecords] [-c logupdates (tableNames)] [-g (tableNames)] [-s (tableNames)]  
[-lcfg logging configuration filename]
```

In actual use, the command must be entered on one line. Also, you must either change to the directory in which the program is stored (typically, \LODESTAR\bin) before entering the command, or specify the path in the command. To view a list of all parameters on-screen, type **plimport -?** at the command prompt.

Parameter	Description
-f	<i>filename</i> is the name of the input file (required for import). Can be either a *.imp or *.xml file. If you import a *.xml file, only the -f, -u, -p, -d, -q, and -e parameters are used. All other parameters are ignored. However, several of these “ignored” settings can be specified within the XML import file. See Appendix B: Oracle Utilities Data Repository Schema Import File Format for more information.

Parameter	Description
-d	<p><i>connectstring</i> is database connection information for the Oracle Utilities Data Repository. This parameter is required and must be in one of the following formats:</p> <p>For Oracle databases:</p> <pre>"Data Source=<data_source>;User ID=<user_id>;Password=<password>;LSProvider=ODP;"</pre> <p>For Microsoft SQL databases:</p> <pre>"Data Source=<address>;Initial Catalog=<SQL_database>;User Id=<user_id>;Password=<password>;MultipleActiveResultSets=True ;LSProvider=MSSQL;"</pre> <p>where:</p> <ul style="list-style-type: none"> • <data_source> is the Oracle TNS Name for the data source, from the TNS_NAMES.ora file (typically located in the \\<machine>\oracle\network\admin directory) • <user_id> is the user ID for the database connection • <password> is the password for the supplied user ID. • <address> is the IP address or Hostname of the MS SQL Server database server • <SQL_database> is the name of the MS SQL Server database
-q	<i>qualifier</i> is an optional database qualifier. The default is PWRLINE.
-m	<p><i>mode</i> is any one of (1, 2, 3, or 4) where:</p> <ul style="list-style-type: none"> • 1 - ADD. Adds NEW records to the Data Repository. If any records in the file already exist in the database, or if there are duplicate record IDs in the Import file, the program considers it an error. • 2 - UPDATE. Replaces EXISTING records in the Data Repository with the version in the Import file. If any records in the file do not already exist in the database, or if there are duplicate record IDs in the Import file, the program considers it an error. • 3 - DELETE. Deletes records from the Data Repository. For every parent ID in the Import file, the program automatically deletes both the parent and all of its dependents. If any records in the file do not already exist in the database, or if there are duplicate record IDs in the Import file, the program considers it an error. • 4 - ADD or UPDATE. Adds NEW records and replaces EXISTING records. If a record is not already in the database, it is added. If it does exist, it is updated. <p>Mode is required for import.</p>

Parameter	Description
-t	<p><i>tablename</i> allows you to view tables in the database. If you want to view the entire database, enter -t without the tablename parameter. If you want to view a selected table, enter -t and the “real” database tablename. To find this tablename, open Data Manager and select File-›Browse-›Database Schema. The “real” tablename is the second one in the label to the right of the table icons (for example, Accounts / ACCOUNT/ ACCT). The second name, ACCOUNT, is the tablename required for this use. See Chapter 6: Browsing Data for additional information about using the Browser to view the database layout.</p> <p><i>Note:</i> The tablename is case sensitive; if you use it, you must enter it in UPPERCASE.</p> <p>Also, using the -t switch overrides the other parameters. PLIMPORT can either import data or display tables in the database, but not both in the same command line. In other words, if you use the -t switch in the command line, PLIMPORT will NOT import data.</p>
-i	<p><i>configfilename</i> is the name of the configuration file that defines the working environment of the Oracle Utilities software (directs the software where to find and place the application data files, and so on). If you do not supply a value for <i>configfilename</i>, the system uses the default (LODESTAR.CFG). For information about the contents of this configuration file, see the <i>Oracle Utilities Energy Information Platform Configuration Guide</i>.</p>
-l	<p><i>logfilename</i> specifies a non-default name for the log file (default name is “plimport.log”).</p>
-e	<p><i>errorfile</i> is the name of an optional file that will contain any invalid records from the original Import file. If the program encounters any invalid records during processing, it copies them as is into the error file. You can correct the records in the error file, and then use it as the new Import file in a second run of the program. If you are importing a large quantity of data, you may have to repeat this process several times, each time whittling down the number of errors until all of the data is imported successfully. In that case, you may want to use a file naming convention to help keep track of the file sequence; e.g., file1.imp, file2.imp, file3.imp, and so on. If you use the same name and path for each run, the program overwrites the file’s contents. If you supply the switch but no errorfile name, the program uses the default name, plimport.err. If you do not supply a path, the error file is placed in the LODESTAR\USER directory.</p>
-x	<p><i>maxerrors</i> is an optional argument that indicates to stop processing after the given maximum number of errors is reached. If you do not include this argument, the program processes the entire file, regardless of the number of invalid records in it. Invalid records are not entered into the database.</p>
-b	<p><i>batchsize</i> is the number of records accumulated in a batch file before the program commits them to the database. Use this setting to optimize the performance (speed) of the Import process. The default value is 1 record. If performance is slow and you have spare hard disk space and memory capacity, try increasing the number.</p>

Parameter	Description
-a	<i>logrecords</i> If you supply this switch, the program logs every record processed, whether it had an error or not.
-c	<i>logupdates</i> indicates that updated fields be logged—the log will show the old and new values for every field whose value changed. Because this option can significantly slow processing time, you may want to shorten the log by additionally supplying a list of table names, separated by comma and enclosed in parentheses; the utility will log only records that were added or updated in the specified tables.
-g	indicates that the identify fields be logged. Because this option can significantly slow processing time, you may want to shorten the log by additionally supplying a list of table names, separated by comma and enclosed in parentheses; the utility will log only records that were added or updated in the specified tables.
-s	<i>tablenames</i> indicates that any previous records' STOPTIME needs to be updated for tables in "tableNames". "tableNames" is a list of table names, separated by commas and enclosed in ""; the utility will log only records that were added or updated in the specified tables.
-lcfg	<i>logging configuration filename</i> Name of an optional logging configuration file that specifies where error and log messages are sent. If you omit this parameter, the application creates a log file named PLIMPORT.LOG in the LOG directory.

Example

The following example would display the layout of the Account Table in a database with the data source name "LODESTAR":

```
PLIMPORT -d LODESTAR -t ACCOUNT
```

Notice that the tablename (ACCOUNT) is in UPPERCASE.

Printing the Database Table Layouts

You can use the `-t` switch in PLIMPORT to view the layout of the Data Repository on your screen. You can also use PLIMPORT and the `-t` switch to send this information to a text-only output file, which you can save for later viewing and printing.

Note: Using the `-t` switch overrides the other parameters. PLIMPORT can either import data or display tables in the database, but not both in the same command line. In other words, if you use the `-t` switch in the command line, PLIMPORT will NOT import data.

How to print the database table layouts:

1. Open a DOS or Windows NT Command Window. (In Windows NT, click the Windows **Start** button, point to **Run**, then type 'cmd' in the field that appears.)
2. Change to the directory in which your Oracle Utilities executables are stored, typically \LODESTAR\BIN.
3. At the prompt, type the following command:

```
PLIMPORT -d dsn -u userid -p password -t TABLENAME > outputfile.txt
```

The `-d`, `-u`, `-p`, and `-t` switches are identical to the switches described previously in this chapter, except that the parameter to the `-t` switch determines which table(s) are *written* to the output file, rather than *displayed* on screen.

outputfile.txt is the name you assign to the output file. The .txt extension is recommended.

For example, the following command would write the layout of the Account Table in the database "lodestar" to a file named acctpict.txt:

```
PLIMPORT -d lodestar -t ACCOUNT > acctpict.txt
```

After you enter the command, PLIMPORT writes the text-only output file containing the layout of the specified table(s) to the current directory. You can view and print the file using Notepad or another text editor.

Importing Interval Data

You can import interval data in batch mode using the INTDIMP.EXE command line program.

Inputs

Before you run INTDIMP, ensure that the following items are in place:

- **Import file:** File containing the raw data to be imported. It can be in any of the following formats:
 - Oracle Utilities Enhanced Database Format (Pervasive.SQL) (*.bte)
 - Enhanced Oracle Utilities Input/Output Format (*.lse)
 - Oracle Utilities Standard Format (*.inp)
 - Oracle Utilities Comma Separated Format (*.csv)
 - Oracle Utilities Standard XML Format (*.xml)
 - MV90 Mainframe Format (*.mv9)
 - EU Engineering Units Format (*.eu)
 - DTV Engineering Units Format (*.dtv)
 - Meter Data Exchange Format (*.mdf).

See **Appendix C: Oracle Utilities Enhanced Input/Output Interval Data Format**, **Appendix D: Oracle Utilities Standard XML Interval Data Format**, and **Appendix E: Oracle Utilities Comma Separated Values (CSV) Interval Data Format** in the *Oracle Utilities Energy Information Platform Configuration Guide* for more information about the interval data formats.

- **Customer Records:** (*Optional*) The entities the data belongs to (accounts, ESPs, etc.) should be defined in the Data Repository down through the Recorder, Channel, and Channel History record levels. **Note:** If the Customer records do not exist, you can still import the cuts, but you will be unable to access the cuts for billing and other purposes, because they will not be associated with an account or other entity.

Important Note

All users should avoid working with the Browser while the import is taking place. Otherwise, the Browser and the Import utility may conflict and contend for system resources.

INTDIMP Command Syntax

The command to run INTDIMP uses the syntax shown below. Parameter switches are case insensitive; you can enter them in either upper or lower case (-c or -C). If a parameter includes a space, you must enclose it in quotes (for example, -s "11/01/1999 12:00:00").

```
intdimp -i importfilename -o outputfilename [-f configfilename] [-e exportfilename]
[-c filename] [-l logfile] [-v] [-d connectstring [-q qualifier]] [-h suppress error log]
[-z timezoneconversion] [-w export validation flag] [-lcfg logging configuration filename]
```

In actual use, the command must be entered on one line. Also, you must either change to the directory where the program is stored (typically, \LODESTAR\bin) before entering the command, or specify the path in the command. To view a list of all parameters on-screen, type **intdimp -?** at the command prompt.

Parameter	Description
-i	<p><i>importfilename</i> is the name of the interval data file to be imported. Input file extensions and file types include:</p> <ul style="list-style-type: none"> .lse - Oracle Utilities Enhanced Format File, .inp - Oracle Utilities Standard Format File, .mdf - Meter Data Exchange Format .mv9 - MV-90 Mainframe File Format .dtv - Engineering Units Format Files .eu - Engineering Units Format Files .csv - Comma Separated Value File Format .xml - Oracle Utilities Standard XML Format File
-o	<p><i>outputfilename</i> is the name of the Interval Database where the data will be input. Output file extensions and file types include:</p> <ul style="list-style-type: none"> .bte - Oracle Utilities Enhanced Btrieve Interval Data, .csv - Oracle Utilities CSV Interval Data, .xml - Oracle Utilities XML Interval Data, <p>Alternatively, you can specify one of the following to save the data to the Oracle Utilities Data Repository:</p> <ul style="list-style-type: none"> RDB (no file name) - default interval data table selected on the Interval Data Sources tab on the Default Options dialog (see Interval Data Source on page 2-12), or RDB/<alternate_table> - a specific table (other than the default table) where: <p><alternate_table> is a string containing the name of a table with the same schema as the LSCHANNELCUTHEADER table. The name of this table must begin with the letters "LSC". Also, this table must have two child tables, one with column VALUECODES and one with column SEQUENCE, and which have the same schema as the LSCHANNELCUTDATA and LSCHANNELCUTEDITS tables, respectively. This table must also have one parent table, the CHANNEL table, which in turn also has one parent table, the RECORDER table.</p> <p>If this parameter is not specified, the data is imported into the LSCHANNELCUTHEADER table.</p>

Parameter	Description
-f	<i>configfilename</i> is the name of the Configuration file that defines the working environment of the Oracle Utilities products (tells the application where to find and place the application data files, etc.). If you do not supply a value for <i>configfilename</i> , the system uses the default (LODESTAR.CFG). For information about the contents of this Configuration file, see the <i>Oracle Utilities Energy Information Platform Configuration Guide</i> .
-e	<p>This switch is optional. If you supply it, the program automatically exports any interval data cuts that are about to be overwritten by the imported cuts. If you supply this switch, you must also supply the full path and name of the Export file (including the .bte, or .inp extension).</p> <p>If you enable this option, the program identifies any cuts in the Interval Database with the same recorder-id, channel-number, and start-time as a cut in the Import file. It compares the cuts' stop time, SPI, UOM, start and stop readings, and pulse and meter multiplier and offsets; as well as their interval values and status codes, interval by interval. If there are differences, the application exports the old cut—appending it to the contents of the export file—before writing the new version to the database.</p> <p>Note: The Key of the cut (the Recorder ID, channel, and start-times) must match exactly. Overlapping cuts that do not have matching Keys are ignored.</p> <p>Enabling this option can slow the Import process considerably.</p>
-c	<p>This switch enables the “Check import data against CHANNELHISTORY values” option. If you supply this switch, the Import utility compares the import cut's header information with the description of the recorder that's stored in the Channel History Table (pulse multiplier, pulse offset, unit of measure, and IPH). If there is a mismatch, the cut fails and is not imported. If a null value is present in either the header or the Channel History record where the other record has a descriptive value, this does not cause the cut to fail (only contradictory information causes an error). If you do not supply this option, no comparison is made.</p> <p>If you supply the -c switch and the optional filename, the program writes the cuts that fail to the specified file.</p>
-l	<p><i>logfile</i> is the name of the optional log file used for output of error messages and additional information about the program's execution. If you do not use this parameter, the default is INTDIMP.RPT, which is placed in your USER directory.</p> <p>To flush the contents of this file after each time the file is written to, use -lf.</p>
-v	This optional switch indicates that data overlap will be checked (if MV90 extension present). The default is no overlap check.

Parameter	Description
-d	<p><i>connectstring</i> is database connection information for the Oracle Utilities Data Repository. This parameter is required and must be in one of the following formats:</p> <p>For Oracle databases:</p> <pre>"Data Source=<data_source>;User ID=<user_id>;Password=<password>;LSPProvider=ODP;"</pre> <p>For Microsoft SQL databases:</p> <pre>"Data Source=<address>;Initial Catalog=<SQL_database>;User Id=<user_id>;Password=<password>;MultipleActiveResultSets=True;LSP rovider=MSSQL;"</pre> <p>where:</p> <ul style="list-style-type: none"> • <data_source> is the Oracle TNS Name for the data source, from the TNS_NAMES.ora file (typically located in the \\<machine>\oracle\network\admin directory) • <user_id> is the user ID for the database connection • <password> is the password for the supplied user ID. • <address> is the IP address or Hostname of the MS SQL Server database server • <SQL_database> is the name of the MS SQL Server database
-q	<i>qualifier</i> is an optional database qualifier. The default is PWRLINE.
-h	<i>suppress error log</i> The application automatically creates an error log in the USER directory. Including this switch stops the creation of the error log.
-z	<p>Performs a timezone conversion upon import. This must be in the format: METER_TZCODE/TARGET_TZCODE:CONTROL_FILE.</p> <p>Where:</p> <ul style="list-style-type: none"> • METER_TZCODE is a timezone code the for input time zone. • TARGET_TZCODE is a timezone code the for target time zone. • CONTROL_FILE is an optional file that contains recorder,METER_TZCODE,TARGET_TZCODE triplets that override the command line codes for its recorder. <p>Both timezone codes must be present, or both absent. See Timezone Codes for a list of available timezone codes.</p>

Parameter	Description
-w	<p><i>export validation flag</i> specifies how to set the Validate_Record_Flag on each cut in the import file. Can be one of the following:</p> <ul style="list-style-type: none"> v - Force Valid: Forces the Validate_Record_Flag on each incoming cut to be set to “N”. i - Force Invalid: Forces the Validate_Record_Flag on each incoming cut to be set to “Y” (Default). u - Use Flag: Uses the Validate_Record_Flag of the incoming cut. <ul style="list-style-type: none"> If this is set to “N”, the INTERNAL_VALIDATION, EXTERNAL_VALIDATION, MERGE, and DELETE flags are all set “Y”. If this is set to “Y”, the INTERNAL_VALIDATION, EXTERNAL_VALIDATION, MERGE, and DELETE flags are all set “N”.
-lcfg	<p><i>logging configuration filename</i> Name of an optional logging configuration file that specifies where error and log messages are sent. If you omit this parameter, the application creates a log file named INTDIMP.LOG in the LOG directory.</p>

Example

Here is a sample INTDIMP command:

```
INTDIMP -i myfile.inp -o mydatabase.bte
```

Return Codes

- 0 - All cuts successfully imported.
- 1 - Input error.
- 2 - Initialization error.
- ##9 - Some cuts were bad.
- #9# - Some cuts had Read or Save errors.
- 9## - Other error during import.

Return codes are not displayed on screen, in the logfile, or in the error log but can be used to trigger additional processing.

Mapping MV-90 Data to the Oracle Utilities Data Repository

This section describes the mappings from MV-90 files to the Oracle Utilities Data Repository. MV-90 field names are used in the following descriptions.

Overlap

The 'Check import data overlap option' on the **Interval Data Import Utility** dialog allows you to indicate that overlaps are an error. If you select this option, the dates for each cut are compared to the dates on other cuts for the same recorder and channel in the interval data repository. If there is any overlap it will be an error and the group will be skipped. If you select this and there is a gap, a warning message will be written to the log file, but it is not an error.

Header

Some information in the LODESTAR Interval Data Repository header is not supplied in the MV-90 file. Most of these values that are not supplied will be filled in with the default values, as listed in **Interval Data Repository Header Data** (p. 19). Two values must be supplied by the user when importing data. An option to the program indicates the DSTPARTICIPANT status and the time zone of the input data. The **Interval Data Import DST Option** on page 8-20 section describes this option.

The field mappings from MV-90 Channel Header record to Interval Data Repository Header are:

MV-90 Field Name	LODESTAR Field Name
DC_RECID	RECORDER_ID
LOGCHAN, DC_LOGCHAN	CHANNEL
TA_START	START_TIME
TA_STOP	STOP_TIME
STRTMTR	METER_START_READING
STOPMTR	METER_STOP_READING
DC_MMULT	METER_MULTIPLIER
DC_PMULT	PULSE_MULTIPLIER
DC_POFFS	PULSE_OFFSET
DR_INPHR	SECONDS_PER_INTERVAL (3600 / DR_INPHR)
DC_UMCODE	UOM
STATUS	ACCEPT_REJECT_STATUS
DR_TDATE	TRANSLATION_DATE
DC_FLOW	DC_FLOW
DR_NOTE	DESCRIPTOR (from MV-90 CUSTOMER record)

Oracle Utilities identifies a cut by its recorder ID, channel number and start time. The recorder ID will come from the DC_RECID field in the Channel Header Record. It is treated as a character string, so leading and trailing spaces will be removed, but leading and trailing zeros will be retained. The channel number is a three-digit number. The low order two digits (00 - 99) will come from the DC_LOGCHAN field, the high order digit (0## through 9##) will come from the LOGCHAN field, both in the Channel Header Record. The start time comes from the TA_START field in the Channel Header Record.

Note: The MV-90 DR_INPHR field must NOT be set to '00.'

Values

The MV-90 DATA_TYPE field will be used to determine if values are in pulses or engineering units (EU). The two valid values for this field are 'P' or 'E' (Pulse or Engineering). Any other value will be an error. All Oracle Utilities interval data values are stored as engineering units. MV-90 pulse values will be converted to engineering units using the formula:

$$EU = (\text{pulse} * DC_PMULT) + DC_POFFS$$

Status Codes

When interval data in MV90 format is imported into the Oracle Utilities Data Repository, any Channel or Interval status codes in the incoming data are mapped to Oracle Utilities status codes based on the configuration of the MV90 Status Code Map table. MV90 status codes are set at specific bits within the MV90 format. The mapping for each bit in the MV90 Status Code table determines the resulting Oracle Utilities status code.

MV90 data can store multiple status codes for the channel and for each individual interval. Oracle Utilities stores only individual status codes for each interval (see Note below). If more than one bit is set on the same interval, the mapping with the higher priority is used.

Example: When importing MV90 data with interval status codes set at bits 1, 2, and 3 for a single interval, the following records in this table would result in a Oracle Utilities status code of 4 for that interval.

Mapping 1

- MV90 Status Word: Interval Status
- Bit: 1
- LODESTAR Status Code: 2
- Priority: 1

Mapping 2

- MV90 Status Word: Interval Status
- Bit: 2
- LODESTAR Status Code: 3
- Priority: 2

Mapping 3

- MV90 Status Word: Interval Status
- Bit: 3
- LODESTAR Status Code: 4
- Priority: 3

Note: The MV90 Status Code Map table is used when mapping MV90 status codes to standard Oracle Utilities status codes only. This table is NOT used to define Oracle Utilities Extended Status Codes.

Relational Data Import

Some values in the MV-90 Customer Header and Channel Header correspond to column values in the Oracle Utilities Data Repository. The actual table and column that corresponds to a MV-90 field depends on the clients database configuration. The action to take when a record or value is missing or different from a MV-90 field value also depends on the client. The following is intended to explain how MV-90 data is mapped into their Oracle Utilities Data Repository.

A MV-90 field may map to zero, one or more database columns (though only one column per table). In some cases, you may want a MV-90 field change to update one column, but add a new record for another table. In addition, you may want a change that causes a mismatch to generate an error. The following rules address these issues, and are used when importing relational data:

- a) If record is not available, create it,
- b) If record is not available, error,
- c) If record is available and corresponding column equals field, do nothing,
- d) If record is available and corresponding column is NULL, update column,
- e) If record is available and corresponding column does not equal field, error,
- f) If record is available and corresponding column does not equal field, update column,
- g) If record is available and corresponding column does not equal field, stop current record, add new one.

Rule c is always in effect.

Several MV-90 fields may be required to key a database record. However, in the Oracle Utilities Data Repository, relative to interval data, all record keys can be derived from recorder id, channel and start time. In the following, when a table is specified its key is automatically derived from these three values.

The import program uses a mapping file - MV90MAP.TXT - to determine how to map MV-90 data to the Oracle Utilities Data Repository. This file is **REQUIRED** to run the MV-90 import program. See **Sample Mappings** on page 8-18 for a sample version of this file. The file must be an ASCII text file. The first line indicates the input file type and consists of the keyword INPUT followed by a comma and the keyword MV90.

The filename and path must be specified using the MV90MAPFILE config parameter:

```
MV90MAPFILE=\\FILESERVER\PATH\MV90MAP.TXT
```

If the parameter is specified with no value, then the import is done without relation database update. If the parameter is not specified then any use of an MV9 file extension will produce an error indicating that the feature is not enabled.

Each subsequent line describes one mapping - the MV-90 record and field, the Oracle Utilities Data Repository table and column (virtual column), and the action, comma separated. Spaces around commas are ignored. There are three MV-90 records - CUSTOMER, CHANNEL, and DEFAULT. The action can be one of four letters - E, A, U or (Error, Add, Update or Insert). Action I can only apply to the DEFAULT record (that is, if the record is DEFAULT, then its action can not be anything but I). Action E, A, and U can apply to mappings from any record but DEFAULT. Blank lines and lines that start with a semi-colon (;) are ignored.

If the MV-90 record is DEFAULT, then the value in the MV-90 field may be used as the column value. It is used only when adding a new record to the database and the corresponding column in the database does not have a value in it.

Mapping must be put in order. Parents table must be mapped before its dependent tables.

Every column in the Oracle Utilities table's identity must be specified, except for the STARTTIME column, which is described below.

If two fields and their corresponding column values are both not equal, the action with the highest priority is taken. The priority order, from high to low, is Error, Add and Update. Thus if one field says Update and another says Add, a new record will be created with both new values - their values in the original record will be unchanged.

If a record is added because of a mis-compare with an existing record, all unchanged columns in the original record will be copied to the new record.

The action I is used only if a record is added, the corresponding columns are not compared. It is not used if there was an old record, the old record's values are always used.

In the special case of DC_LOGCHAN as the MV-90 field, the LOGCHAN field will also be used to create a three digit channel number, just as in the Interval Data Import section above.

The record in a table with a STARTTIME and STOPTIME column is chosen as the last one in effect on the cut stop (TA_STOP). If there is no overlap this means the record is not available (see Rules a and b above). If the table contains a STOPTIME column and an Add action happens, the original record's STOPTIME will be updated to be either the previous cut's stop time (if overlap check is on) or the new record's STARTTIME minus one second.

The lines for recorder and channel should always be first in the file (recorder first, channel second). The only actions allowed for them are Add and Error (see Rules a and b above). Records in all other tables are found via the straight path from the RECORDER and CHANNEL tables.

The MV-90 Customer record will be processed once, using the first channel header's recorder and channel. All Channel records will be processed once per channel header record.

The order of the file can affect performance, since the first error will cause the import of the cut to stop.

It is an error if a specified field is missing (all binary zeros or spaces) in the MV-90 cut and is required for the relational database. If it is missing and not required, the corresponding field will be set to NULL.

It is an error if a Oracle Utilities Data Repository table or column does not exist. It is an error if the same table and column is used to match more than one MV-90 field. This validation happens before any data is imported.

Sample Mappings

The following is a sample MV90MAP.TXT file:

```
; Input type
INPUT,MV90

;      MV-90                      LODESTAR Customer Database
;RecordFieldTableColumn Action

DEFAULT,TVA,OPERATINGCOMPANY,OPCOCODE,I
DEFAULT,TVA,OPERATINGCOMPANY,OPCONAME,I

DEFAULT,Tennessee,JURISDICTION,JURISCODE,I
DEFAULT,Tennessee,JURISDICTION,JURISNAME,I

CHANNEL,DC_RECID,RECORDER,RECORDERID,E
DEFAULT,10/10/1980,RECORDER,STARTTIME,I

CHANNEL,DC_LOGCHAN,CHANNEL,CHANNELNUM,E

CUSTOMER,CM_ADDR1,SUBSTATION,SUBSTATIONNAME,U
DEFAULT,TVA,SUBSTATION,OPCOCODE,I
DEFAULT,TVA,SUBSTATION,JURISCODE,I

CUSTOMER,CM_ADDR2,TRANSFORMERLOC,TRANSFORMERLOCNAME,U

CHANNEL,DC_RECID,CHANNELHISTORY,RECORDERID,A
CHANNEL,DC_LOGCHAN,CHANNELHISTORY,CHANNELNUM,A
CHANNEL,DC_UMCODE,CHANNELHISTORY,UOMCODE,A
CHANNEL,DR_INPHR,CHANNELHISTORY,IPH,A
CHANNEL,DC_MMULT,CHANNELHISTORY,METERMULTIPLIER,A
CHANNEL,DC_PMULT,CHANNELHISTORY,PULSEMULTIPLIER,A
CHANNEL,DC_POFFS,CHANNELHISTORY,PULSEOFFSET,A
CHANNEL,DC_PTRATIO,CHANNELHISTORY,VT,A
CHANNEL,DC_SERVTYPE,CHANNELHISTORY,SERVTYPE,A (not converted to # wires)
CHANNEL,DC_NDIALS,CHANNELHISTORY,NOOFDIALS,A
CHANNEL,DC_METERID,CHANNELHISTORY,METERNUMBER,A
DEFAULT,Y,CHANNELHISTORY,BILLED,I
DEFAULT,I,CHANNELHISTORY,TOTALIZE,I

CHANNEL,DR_DEVSN,RECORDERHISTORY,SERIALNUMBER,A
DEFAULT,10/10/1980,RECORDERHISTORY,INSTALLDATE,I
```

The first CHANNELHISTORY line will cause a new record to be created if there is no corresponding record.

Logging and Error Messages

A log file is generated during the import process. It describes each cut imported and provides specific error messages if an error occurs. The error message indicates the recorder and channel that failed, with as much detail about the error as is known.

A second log is also written. It's name is LGddmmyy.LSI (dd = day, mm = month, yy = year, LSI for LODESTAR Import). Only one file is created per day, subsequent runs during the day will append to the day's file. There will be one line per customer record and per MV-90 channel header processed.

A channel header record consists of the input file name, RCODE 10, eight digit channel header line number, 20 character recorder id (DC_RECID), three digit channel number (LOGCHAN DC_LOGCHAN), start time (MMDDYYHHMM from TA_START), stop time (MMDDYYHHMM from TA_STOP) and a success code, all comma separated, all values from the Channel Header Record. The success code is zero for successful import and one for any error

or failure, even of another cut in the same group. When an error happens to a cut and the group is skipped, a log message is still written for every cut in the group.

A customer record consists of the input file name, a two digit RCODE 1, an eight digit customer header line number, a 20 character customer id (CM_CUSTID), the number of expected channel records (CM_LOGCHANS, three digits), start time (MMDDYYHHMM from TA_START), stop time (MMDDYYHHMM from TA_STOP) and a success code. All the fields are comma separated, and come from the Customer Header Record. The success code is zero (0) if all cuts imported successfully, and is one (1) for any error or failure of any cut following this customer record and before another customer record.

Note that all fields after the input file are fixed width. Numeric fields are right justified with leading zeros replaced with spaces. Character fields are left justified and blank filled. If the input file names are all the same length, the file contains records with all fixed width columns.

An example of the log file is as follows:

```
JAN01.MV9,10,      2,1013001      , 1,0215990804,0314990810,0
JAN01.MV9,10,     100,1013001     , 2,0215990804,0314990810,0
JAN01.MV9,10,     200,1013001     , 5,0215990804,0314990810,0
JAN01.MV9, 1,      1,1013001      , 3,0215990804,0314990810,0
JAN01.MV9,10,     321,6833057     , 1,0215990820,0314990830,1
JAN01.MV9,10,     442,6833057     ,124,0215990820,0314990830,1
JAN01.MV9, 1,      320,6833057     , 2,0215990820,0314990830,1
```

Interval Data Repository Header Data

The following data is stored in each record (cut) in the Oracle Utilities Data Repository:

Field Name	Data Type	Default Value
RECORD_TYPE	char	' ' (space)
RECORDER_ID	char	
CHANNEL	integer	
START_TIME	datetime	
DATE_TIME_STAMP	char	Date and time of import
STOP_TIME	datetime	
METER_START_READING	float	
METER_STOP_READING	float	
METER_MULTIPLIER	float	
METER_OFFSET	float	0
PULSE_MULTIPLIER	float	
PULSE_OFFSET	float	
POPULATION	float	0
WEIGHT	float	0
CALC_CONSTANT	float	0
SECONDS_PER_INTERVAL	integer	3600 / Intervals-per-hour
TOTAL_INTERVALS	integer	computed
BASIC_UNIT_CODE	integer	0
UOM	integer	
TIME_ZONE_WEST_OF_GMT	integer	Import Program input
DATA_TYPE	char	' ' (spaces)
DESCRIPTOR	char	
EDITED	char	' ' (space)
INTERNAL_VALIDATION	char	' ' (space)
EXTERNAL_VALIDATION	char	' ' (space)
MERGE	char	' ' (space)
DELETE_FLAG	char	' ' (space)
DATA_STATUS_FLAG	char	'Y'
START_OFFSET_FLAG	char	' ' (space)
ORIGIN	char	'M' (Metered)

VAL_FLAG_E	char	' ' (space)
VAL_FLAG_I	char	' ' (space)
VAL_FLAG_O	char	' ' (space)
VAL_FLAG_N	char	' ' (space)
TK_WRITTEN_FLAG	char	'N'
DST_PARTICIPANT	char	Import Program input

New interval data cut fields

Field Name	Data Type	Default Value
MV90_STATUS	char(2)	Cut Accept/Reject Status Code
MV90_DR_TDATE	datetime	Translation Date / Time
MV90_DC_FLOW	char(1)	D or ' ' (Delivered), R (Received)

Interval Data Import DST Option

All interval data is stored with the time adjusted to match the accounts that use it. An option on the interval data import dialog and command line (the DST button) allows the user to specify the timezone of the input file and the timezone of its target accounts.

TIMEZONE Description

A timezone code describes the timezone. Four fields are associated with a timezone:

- **CODE** Usually three or four letters such as EDT or EDTA, EST, CST, ...
- **NAME** Visible name - Eastern Daylight Time, Eastern Standard Time, Central Standard Time,
- **TIMEZONE** An integer that is the number of timezones west of GMT (Greenwich Mean Time).
- **DST**
 - Y - Indicates that all dates and times reflect Daylight Savings Times, 23 hours in spring, 25 in fall.
 - A - Indicates that all dates and times reflect Daylight Savings Times, intervals are 24 hour adjusted.
 - N - Standard time.

Some examples:

CODE	NAME	TIMEZONE	DST
EST	Eastern Standard Time	5	N
EDT	Eastern Daylight Time	5	Y
EDTA	Eastern Daylight Time - Adjusted	5	A

See **Timezone Codes** on page 8-21 for a complete listing of Time Zone codes.

Interval Data Import

The Interval Data Import dialog has a DST dialog that lets the user input the meter and target timezones when importing interval data. This is the **Timezone/Daylight Savings Time Options** dialog. In addition, the value supplied to the -z switch on INTDIMPEXE is the meter timezone, a slash, then the target timezone, and then an optional colon and recorder specific control file.

The meter timezone specifies how the data was measured, the target timezone specifies how the data is to be stored. The recorder specific control file contains a list of recorders and their individual meter and target timezones, comma separated (one recorder per line). Spaces around the commas are ignored. The values in the control file override the command line or user selected meter and target timezones. The command line meter and target timezones must both be present, or both missing.

A sample control file might look like this:

```
RECORDER1,EST,EDT
RECORDER2,EDTA,EDT
```

Sample command lines are:

intdimp ...	Import all cuts as they are
intdimp ... -zEST/EDT	Import and convert all cuts from EST to EDT
intdimp ... -zEST/EDT:PSI.TXT	Convert all cuts from EST to EDT, unless otherwise specified in PSI.TXT
intdimp ... -z:PSI.TXT	Import all cuts as they are, unless otherwise specified in PSI.TXT

If an input format is used that supports meter timezone information (such as the LodeStar enhanced format) and the values in the input file are different from the input meter option an error will be generated.

If the expected count of intervals differs from that in the file, an error will be generated. The expected count is basically determined by this equation (where times are measured in seconds):

$$\text{Interval count} = ((\text{stop-time} + 1) - \text{start-time}) / \text{second-per-interval},$$

adjusted by plus or minus the IPH if the dates cross a DST transition (first Sunday in April, last Sunday in October).

Timezone Codes

CODE	NAME	TIMEZONE	DST
GMT	Greenwich Mean Time	0	N
BST	British Summer Time	47	N
BSTA	British Summer Time - Adjusted	0	A
IST	Irish Summer Time	47	N
ISTA	Irish Summer Time - Adjusted	0	A
WET	Western Europe Time	0	N
WEST	Western Europe Summer Time	0	Y
WESTA	Western Europe Summer Time - Adjusted	0	A
CET	Central Europe Time	46	N
CEST	Central Europe Summer Time	46	Y
CESTA	Central Europe Summer Time - Adjusted	46	A
EET	Eastern Europe Time	44	N
EEST	Eastern Europe Summer Time	44	Y
EESTA	Eastern Europe Summer Time - Adjusted	44	A
MSK	Moscow Time	42	N
MSD	Moscow Summer Time	42	Y

CODE	NAME	TIMEZONE	DST
MSDA	Moscow Summer Time - Adjusted	42	A
AST	Atlantic Standard Time	8	N
ADT	Atlantic Daylight Time	8	Y
ADTA	Atlantic Daylight Time - Adjusted	8	A
EST	Eastern Standard Time	10	N
EDT	Eastern Daylight Time	10	Y
EDTA	Eastern Daylight Time - Adjusted	10	A
CST	Central Standard Time	12	N
CDT	Central Daylight Time	12	Y
CDTA	Central Daylight Time - Adjusted	12	A
MST	Mountain Standard Time	14	N
MDT	Mountain Daylight Time	14	Y
MDTA	Mountain Daylight Time - Adjusted	14	A
PST	Pacific Standard Time	16	N
PDT	Pacific Daylight Time	16	Y
PDTA	Pacific Daylight Time - Adjusted	16	A
UTC-12	Universal Time Coordinated - 12 hours	24	N
UTC-12D	Universal Time Coordinated - 12 hours (DST)	24	Y
UTC-12DA	Universal Time Coordinated - 12 hours (24 Hr Adjusted)	24	A
UTC-11	Universal Time Coordinated - 11 hours	22	N
UTC-11D	Universal Time Coordinated - 11 hours (DST)	22	Y
UTC-11DA	Universal Time Coordinated - 11 hours (24 Hr Adjusted)	22	A
UTC-10	Universal Time Coordinated - 10 hours	20	N
UTC-10D	Universal Time Coordinated - 10 hours (DST)	20	Y
UTC-10DA	Universal Time Coordinated - 10 hours (24 Hr Adjusted)	20	A
UTC-9	Universal Time Coordinated - 9 hours	18	N
UTC-9D	Universal Time Coordinated - 9 hours (DST)	18	Y
UTC-9DA	Universal Time Coordinated - 9 hours (24 Hr Adjusted)	18	A
UTC-8	Universal Time Coordinated - 8 hours	16	N
UTC-8D	Universal Time Coordinated - 8 hours (DST)	16	Y

CODE	NAME	TIMEZONE	DST
UTC-8DA	Universal Time Coordinated - 8 hours (24 Hr Adjusted)	16	A
UTC-7	Universal Time Coordinated - 7 hours	14	N
UTC-7D	Universal Time Coordinated - 7 hours (DST)	14	Y
UTC-7DA	Universal Time Coordinated - 7 hours (24 Hr Adjusted)	14	A
UTC-6	Universal Time Coordinated - 6 hours	12	N
UTC-6D	Universal Time Coordinated - 6 hours (DST)	12	Y
UTC-6DA	Universal Time Coordinated - 6 hours (24 Hr Adjusted)	12	A
UTC-5	Universal Time Coordinated - 5 hours	10	N
UTC-5D	Universal Time Coordinated - 5 hours (DST)	10	Y
UTC-5DA	Universal Time Coordinated - 5 hours (24 Hr Adjusted)	10	A
UTC-4	Universal Time Coordinated - 4 hours	8	N
UTC-4D	Universal Time Coordinated - 4 hours (DST)	8	Y
UTC-4DA	Universal Time Coordinated - 4 hours (24 Hr Adjusted)	8	A
UTC-3	Universal Time Coordinated - 3 hours	6	N
UTC-3D	Universal Time Coordinated - 3 hours (DST)	6	Y
UTC-3DA	Universal Time Coordinated - 3 hours (24 Hr Adjusted)	6	A
UTC-2	Universal Time Coordinated - 2 hours	4	N
UTC-2D	Universal Time Coordinated - 2 hours (DST)	4	Y
UTC-2DA	Universal Time Coordinated - 2 hours (24 Hr Adjusted)	4	A
UTC-1	Universal Time Coordinated - 1 hour	2	N
UTC-1D	Universal Time Coordinated - 1 hour (DST)	2	Y
UTC-1DA	Universal Time Coordinated - 1 hour (24 Hr Adjusted)	2	A
UTC	Universal Time Coordinated	0	N
UTCD	Universal Time Coordinated (DST)	0	Y
UTCDA	Universal Time Coordinated (24 Hr Adjusted)	0	A
UTC+1	Universal Time Coordinated + 1 hour	46	N
UTC+1D	Universal Time Coordinated + 1 hour (DST)	46	Y

CODE	NAME	TIMEZONE	DST
UTC+1DA	Universal Time Coordinated + 1 hour (24 Hr Adjusted)	46	A
UTC+2	Universal Time Coordinated + 2 hours	44	N
UTC+2D	Universal Time Coordinated + 2 hours (DST)	44	Y
UTC+2DA	Universal Time Coordinated + 2 hours (24 Hr Adjusted)	44	A
UTC+3	Universal Time Coordinated + 3 hours	42	N
UTC+3D	Universal Time Coordinated + 3 hours (DST)	42	Y
UTC+3DA	Universal Time Coordinated + 3 hours (24 Hr Adjusted)	42	A
UTC+4	Universal Time Coordinated + 4 hours	40	N
UTC+4D	Universal Time Coordinated + 4 hours (DST)	40	Y
UTC+4DA	Universal Time Coordinated + 4 hours (24 Hr Adjusted)	40	A
UTC+5	Universal Time Coordinated + 5 hours	38	N
UTC+5D	Universal Time Coordinated + 5 hours (DST)	38	Y
UTC+5DA	Universal Time Coordinated + 5 hours (24 Hr Adjusted)	38	A
UTC+6	Universal Time Coordinated + 6 hours	36	N
UTC+6D	Universal Time Coordinated + 6 hours (DST)	36	Y
UTC+6DA	Universal Time Coordinated + 6 hours (24 Hr Adjusted)	36	A
UTC+7	Universal Time Coordinated + 7 hours	34	N
UTC+7D	Universal Time Coordinated + 7 hours (DST)	34	Y
UTC+7DA	Universal Time Coordinated + 7 hours (24 Hr Adjusted)	34	A
UTC+8	Universal Time Coordinated + 8 hours	32	N
UTC+8D	Universal Time Coordinated + 8 hours (DST)	32	Y
UTC+8DA	Universal Time Coordinated + 8 hours (24 Hr Adjusted)	32	A
UTC+9	Universal Time Coordinated + 9 hours	30	N
UTC+9D	Universal Time Coordinated + 9 hours (DST)	30	Y
UTC+9DA	Universal Time Coordinated + 9 hours (24 Hr Adjusted)	30	A
UTC+10	Universal Time Coordinated + 10 hours	28	N
UTC+10D	Universal Time Coordinated + 10 hours (DST)	28	Y

CODE	NAME	TIMEZONE	DST
UTC+10DA	Universal Time Coordinated + 10 hours (24 Hr Adjusted)	28	A
UTC+11	Universal Time Coordinated + 11 hours	26	N
UTC+11D	Universal Time Coordinated + 11 hours (DST)	26	Y
UTC+11DA	Universal Time Coordinated + 11 hours (24 Hr Adjusted)	26	A

Importing Enhanced Interval Data

Importing enhanced/generic interval data is performed using the INTDIMPEX command line program. This console application imports interval data in the Compact XML format into a specified enhanced/generic interval data table.

Note: INTDIMPEX does NOT create parent records upon import. Parent records must exist in the database prior to importing interval data using INTDIMPEX.

INTDIMPEX Command Syntax

The command to run INTDIMPEX uses the syntax shown below. Parameter switches are case insensitive; you can enter them in either upper or lower case (-c or -C). If a parameter includes a space, you must enclose it in quotes (for example, -s "11/01/1999 12:00:00").

intdimplex -c *connectstring* [-q *qualifier*] -t *tablename* -i *importfile* -r *rulesfile*

In actual use, the command must be entered on one line. Also, you must either change to the directory where the program is stored (typically, \LODESTAR\bin) before entering the command, or specify the path in the command. To view a list of all parameters on-screen, type **intdimplex -?** at the command prompt.

Parameter	Description
-c	<p><i>connectstring</i> is database connection information for the Oracle Utilities Data Repository. This parameter is required and must be in one of the following formats:</p> <p>For Oracle databases:</p> <p>"Data Source=<data_source>;User ID=<user_id>;Password=<password>;LSProvider=ODP;"</p> <p>For Microsoft SQL databases:</p> <p>"Data Source=<address>;Initial Catalog=<SQL_database>;User Id=<user_id>;Password=<password>;MultipleActiveResultSets=True;LSProvider=MSSQL;"</p> <p>where:</p> <ul style="list-style-type: none"> • <data_source> is the Oracle TNS Name for the data source, from the TNS_NAMES.ora file (typically located in the \\<machine>\oracle\network\admin directory) • <user_id> is the user ID for the database connection • <password> is the password for the supplied user ID. • <address> is the IP address or Hostname of the MS SQL Server database server • <SQL_database> is the name of the MS SQL Server database
-q	<i>qualifier</i> is the required database qualifier.
-t	<i>tablename</i> is the name of the enhanced/generic interval data table into which the data will be imported. The tablename must be in UPPERCASE.
-i	<i>importfile</i> is the name of the file containing the data to import. This file must be in the Compact XML format. See Compact XML Format on page 8-27 for more information.

Parameter	Description
-r	<p><i>rulesfile</i> is the name of an alternate file containing interval data configuration parameters. This file should have the same contents and structure as the INTDCONFIG.CFG.XML configuration file. See INTDCONFIG.CFG.XML on page 2-17 of the <i>Oracle Utilities Energy Information Platform Configuration Guide</i> for more information about this file.</p> <p>If specified, this file overrides settings in the INTDCONFIG.CFG.XML file. If not specified, the application uses the INTDCONFIG.CFG.XML file.</p> <p>Note: If -r is specified in the command line without a value, the data is not imported.</p>

Example

Here is a sample INTDIMPEX command:

```
INTDIMPEX -c "Data Source=LSSRV11;User
ID=TR450;Password=password;LSProvider=ODP;" -q tr450 -t LSINTERVALDATA
-i "c:\lodestar\user\intd_import.xml"
```

Compact XML Format

The INTDIMPEX command line import data in the Compact XML format. See **XML Compact Format** on page D-16 in the *Oracle Utilities Energy Information Platform Configuration Guide* for more information about this format.

When using this format with enhanced/generic interval data tables, the RECORDER and CHANNEL attributes are **not** used. They are replaced with the TIMESERIES attribute, which contains the identity of the parent record of the interval data cut to be imported. Also, when using categories, the CATEGORY attribute should contain the category of the data to be imported.

Custom Columns

The Compact XML format also supports custom data elements that correspond to custom columns added to enhanced/generic interval data tables. Custom elements are added using the <CUSTOM> XML element, which is a child of the <HEADER> element. The format of this element is:

```
<CUSTOM <fieldname>=<value>/>
```

where:

- <fieldname> is an attribute that is the name of the custom column
- <value> is the value to be imported into the custom column

The <fieldname> attribute can be repeated for multiple custom columns.

For example, the following <CUSTOM> element includes import values for the CALCGROUP, MAXIMUM, and MINIMUM custom columns:

```
<CUSTOM CALCGROUP="1" MAXIMUM="10" MINIMUM="-10" />
```

Examples:

Below is an example the compact XML format user to import data into an enhanced/generic interval data table that includes the CALCGROUP, MINIMUM, and MAXIMUM custom columns.

```
<cuts>
  <CUTEX>
    <HEADER TIMESERIES="TEST,1" SPI="900" UOM="1" DST="Y" STARTTIME="2006-12-12
00:01:00" STOPTIME="2007-01-10 23:59:59" TDATE="1969-12-31 19:00:00" ORIGIN="C"
DCFLOW="" ARS="" START_READ="0.0" STOP_READ="0.0" POP="0.0" WEIGHT="0.0"
M_OFFSET="0.0" M_MULT="0.0">
      <CUSTOM CALCGROUP="1" MAXIMUM="10" MINIMUM="-10" />
    </HEADER>
    <INTS>
      <I V="16" S="7" />
      <I V="18" S="7" />
      <I V="16" S="7" />
      <I V="18" S="7" />
      <I V="15" S="7" />
      <I V="18" S="7" />
      <I V="16" S="7" />
      .....
      <I V="21" S="7" />
      <I V="20" S="7" />
      <I V="17" S="7" />
    </INTS>
  </CUTEX>
</cuts>
```

Below is a similar example that includes a category.

```
<CUTS>
  <CUTEX>
    <HEADER TIMESERIES="PERCOT,1" CATEGORY="RAW" UOMCODE="01" STARTTIME="2005-
12-03T00:00:00" STOPTIME="2005-12-03T23:59:59" SPI="3600" DST="Y" P_MULT="1.0"
P_OFFSET="1.0">
      <CUSTOM CALCGROUP="1" MAXIMUM="1" MINIMUM="1"/>
    </HEADER>
    <INTS>
      <I V="2850" S=" " />
      <I V="3150" S=" " />
      <I V="2850" S=" " />
      <I V="3150" S=" " />
      <I V="2850" S=" " />
      <I V="3150" S=" " />
      <I V="2850" S=" " />
      <I V="3150" S=" " />
      <I V="2850" S=" " />
      <I V="3150" S=" " />
      <I V="2850" S=" " />
      <I V="3150" S=" " />
      <I V="2850" S=" " />
      <I V="3150" S=" " />
    </INTS>
  </CUTEX>
</CUTS>
```


Exporting Data

This section describes how data can be exported and/or deleted from the Oracle Utilities Data Repository in batch mode. This includes:

- Exporting and Deleting Customer Data
- Exporting and Deleting Interval Data

Exporting and Deleting Customer Data

You can export and delete customer data in batch mode using the EXPCA.EXE and EXPTBL.EXE command line programs. The first is used to delete/export selected customers or accounts; the second is used to delete the contents of a specified table.

Using EXPCA

EXPCA enables you to export and/or delete a customer or account, a list of customers or accounts, or all customers or accounts.

EXPCA Command Syntax

EXPCA uses the syntax shown below. Parameter switches are case insensitive; you can enter them in either upper or lower case (-c or -C). If a parameter includes a space, you must enclose it in quotes (for example, -f “11/01/1999 12:00:00”). Also, when you enter the command, you must either supply the path to the program, or run it from the directory where it is stored (typically, \LODESTAR\BIN).

```
expca -o outputfilename [-d connectstring] [-q qualifier] [-c configfilename]
[-m qualifiers id/ list] [-e expmode] [-f "fromdate"] [-t "todate"] [-k exportlookup]
[-h exportchild] [-r exportparent] [-v overwrite] [-l logfilename]
[-lcfg logging configuration filename] [-x xmloutput]
```

In actual use, the command must be entered on one line. Also, you must either change to the directory where the program is stored (typically, \LODESTAR\bin) before entering the command, or specify the path in the command. To view a list of all parameters on-screen, type **expca -?** at the command prompt.

Parameter	Descriptions
-o	<i>outputfilename</i> is the name of the file the records will be exported to. This can be either a *.imp file or a *.xml file. If you supply a *.xml file, you must also supply the -x switch.

Parameter	Descriptions
-d	<p><i>connectstring</i> is database connection information for the Oracle Utilities Data Repository. This parameter is required and must be in one of the following formats:</p> <p>For Oracle databases:</p> <pre>"Data Source=<data_source>;User ID=<user_id>;Password=<password>;LSPProvider=ODP;"</pre> <p>For Microsoft SQL databases:</p> <pre>"Data Source=<address>;Initial Catalog=<SQL_database>;User Id=<user_id>;Password=<password>;MultipleActiveResultSets=True ;LSPProvider=MSSQL;"</pre> <p>where:</p> <ul style="list-style-type: none"> • <data_source> is the Oracle TNS Name for the data source, from the TNS_NAMES.ora file (typically located in the \\<machine>\oracle\network\admin directory) • <user_id> is the user ID for the database connection • <password> is the password for the supplied user ID. • <address> is the IP address or Hostname of the MS SQL Server database server • <SQL_database> is the name of the MS SQL Server database
-q	<i>qualifier</i> is an optional database qualifier. The default is PWRLINE.
-c	<i>configfilename</i> is the name of the Configuration file that defines the working environment of the Oracle Utilities software (tells the software where to find and place the application data files, etc.). If you do not supply a value for <i>configfilename</i> , the system uses the default (LODESTAR.CFG). For information about the contents of this configuration file, see the <i>Oracle Utilities Energy Information Platform Configuration Guide</i> .
-m	<i>qualifiers</i> a 2-character string where the first character specifies a customer or account ('c' or 'a') and the second character specifies whether it is going to be an ID ('i'), a list ('l'), a name ('n'), or all accounts/customers ('a'). If ID, name, or list, then the actual ID/name/list is the next parameter.
-e	<i>expmode</i> selects the Export mode; 0 (export), 1 (delete), or 2 (export/delete). The default is 0.
-f	" <i>fromdate</i> " is the start of the date range. Dates can be in mm/dd/yyyy hh:mm:ss or yyyy/mm/dd hh:mm:ss format.
-t	" <i>todate</i> " is the end of the date range. Dates can be in mm/dd/yyyy hh:mm:ss or yyyy/mm/dd hh:mm:ss format.
-k	<i>exportlookup</i> If you include this switch, EXPCA exports lookup records to the output file.
-h	<i>exportchild</i> If you include this switch, EXPCA exports child records to the output file.

Parameter	Descriptions
-r	<i>exportparent</i> If you include this switch, EXPCA exports parent records to the output file.
-v	<i>overwrite</i> If you include this switch, EXPCA overwrites the output file. The default is to append the output file.
-l	<i>logfilename</i> is the name of an optional log file, used for output of error messages and additional information about the program's execution. If you do not use this parameter, the default is EXPCA.LOG, which is placed in your USER directory.
-lcfg	<i>logging configuration filename</i> Name of an optional logging configuration file that specifies where error and log messages are sent. If you omit this parameter, the application creates a log file named EXPCA.LOG in the LOG directory.
-x	<i>xmloutput</i> Used if the output file is a *.xml file. This parameter specifies that the output is to be in XML format.

Rules

The following rules apply to EXPCA:

Rules for Export, Delete, and Export/Delete

1. Export lookups (-k option) applies only to export. For delete and export/delete, this option is ignored.
2. Export parent (-r option) applies only to export. For delete and export/delete, this option is ignored.
3. For delete and export/delete, the child records are deleted, even if the '-h' option is not specified.

Date Range Filter Rules

1. The dates can be in "mm/dd/yyyy hh:mm:ss" or "yyyy/mm/dd hh:mm:ss" format. The time (hh:mm:ss) is optional. Mm, dd, yyyy, hh, mm, and ss are numeric.
2. If the from and to dates are supplied, all records whose stop-times fall on or between the supplied dates will be exported and/or deleted.
3. If the to date is supplied with no from date, all records with stop-times on or before the supplied date will be exported and/or deleted.
4. If a from date is supplied with no to date, all records with either a NULL stop-time or a stop-time greater than the supplied date will be exported and/or deleted.
5. If no dates are supplied, all records, regardless of their stop-times, will be exported and/or deleted.

Export and/or Delete Selection Types

The rules for the **-m** *qualifiers id/list* option are:

1. The *qualifiers* will be a 2-character string; where the first character specifies whether it is a customer or account ('c' or 'a'), and the second character specifies whether it is all customers/accounts ('a'), a list of customers/accounts ('l'), or one customer/account ('i').
2. If a customer/account ('i') or a list of customers/accounts ('l') is specified, the customer/account ID or List name must follow (*id/list*).

Examples:

Example Command	Result
-m aa	All accounts. This is the default.
-m ci 800001	The customer "800001".
-m al AcctList1	The account list "AcctList1".

Using EXPTBL

EXPTBL is designed to export, delete, or export/delete a table from your Oracle Utilities Data Repository.

EXPTBL Command Syntax

EXPTBL uses the syntax shown below. Parameter switches are case insensitive; you can enter them in either upper or lower case (-c or -C). If a parameter includes a space, you must enclose it in quotes (for example, -f "11/01/1999 12:00:00").

```
exptbl -o outputfilename [-d connectstring] [-q qualifier] [-c configfilename] -n tablename  
-e expmode [-f "fromdate"] [-t "todate"] [-k exportlookup] [-h exportchild] [-r exportparent]  
[-v overwrite] [-l logfile] [-lcfg logging configuration filename] [-x xmloutput]
```

In actual use, the command must be entered on one line. Also, you must either change to the directory where the program is stored (typically, \LODESTAR\bin) before entering the command, or specify the path in the command. To view a list of all parameters on-screen, type **exptbl -?** at the command prompt.

Parameter	Description
-o	<i>outputfilename</i> is the output file name to export to. This can be either a *.imp file or a *.xml file. If you supply a *.xml file, you must also supply the -x switch. This parameter is not needed for deleting data.

Parameter	Description
-d	<p><i>connectstring</i> is database connection information for the Oracle Utilities Data Repository. This parameter is required and must be in one of the following formats:</p> <p>For Oracle databases:</p> <pre>"Data Source=<data_source>;User ID=<user_id>;Password=<password>;LSPProvider=ODP;"</pre> <p>For Microsoft SQL databases:</p> <pre>"Data Source=<address>;Initial Catalog=<SQL_database>;User Id=<user_id>;Password=<password>;MultipleActiveResultSets=True ;LSPProvider=MSSQL;"</pre> <p>where:</p> <ul style="list-style-type: none"> • <data_source> is the Oracle TNS Name for the data source, from the TNS_NAMES.ora file (typically located in the \\<machine>\oracle\network\admin directory) • <user_id> is the user ID for the database connection • <password> is the password for the supplied user ID. • <address> is the IP address or Hostname of the MS SQL Server database server • <SQL_database> is the name of the MS SQL Server database
-q	<i>qualifier</i> is an optional database qualifier. The default is PWRLINE.
-c	<i>configfilename</i> is the name of the configuration file that defines the working environment of the Oracle Utilities software (tells the application where to find and place the application data files, etc.). If you do not supply a value for <i>configfilename</i> , the system uses the default (LODESTAR.CFG). For information about the contents of the configuration file, see the <i>Oracle Utilities Energy Information Platform Configuration Guide</i> .
-n	<i>tablename</i> is the table name to export from. This is <i>required</i> .
-e	<i>expmode</i> selects the export mode; 0 (export), 1 (delete), or 2 (export/delete). The default is 0.
-f	<i>"fromdate"</i> (Optional) is the start of the date range. Dates can be in mm/dd/yyyy hh:mm:ss or yyyy/mm/dd hh:mm:ss format.
-t	<i>"todate"</i> (Optional) is the end of the date range. Dates can be in mm/dd/yyyy hh:mm:ss or yyyy/mm/dd hh:mm:ss format.
-k	<i>exportlookup</i> (Optional) If you include this switch, EXPCA exports lookup records to the output file.
-h	<i>exportchild</i> (Optional) If you include this switch, EXPCA exports child records to the output file.
-r	<i>exportparent</i> (Optional) If you include this switch, EXPCA exports parent records to the output file.

Parameter	Description
-v	<i>overwrite (Optional)</i> If you include this switch, EXPCA overwrites the output file. The default is to append to the output file.
-l	<i>logfilename (Optional)</i> is the name of an optional log file, used for output of error messages and additional information about the program's execution. If not specified, defaults to EXPTBL.RPT in your USER directory.
-lcfg	<i>logging configuration filename</i> Name of an optional logging configuration file that specifies where error and log messages are sent. If you omit this parameter, the application creates a log file named EXPTBL.LOG in the LOG directory.
-x	<i>xmloutput</i> Used if the output file is a *.xml file. This parameter specifies that the output is to be in XML format.

Rules

The following rules apply to EXPTBL:

Rules for Export, Delete, and Export/Delete

1. Export lookups (-k option) applies only to export. For delete and export/delete, this option is ignored.
2. Export parent (-r option) applies only to export. For delete and export/delete, this option is ignored.
3. For delete and export/delete, the child records are deleted, even if the '-h' option is not specified.

Date Range Filter Rules

1. The dates can be in "mm/dd/yyyy hh:mm:ss" or "yyyy/mm/dd hh:mm:ss" format. The time (hh:mm:ss) is optional. Mm, dd, yyyy, hh, mm, and ss are numeric.
2. If the from and to dates are supplied, all records whose stop-times fall on or between the supplied dates will be exported and/or deleted.
3. If the to date is supplied with no from date, all records with stop-times on or before the supplied date will be exported and/or deleted.
4. If a from date is supplied with no to date, all records with NULL stop-times or stop-times greater than the supplied date will be exported and/or deleted.

If no dates are supplied, all records, regardless of their stop-times, will be exported and/or deleted.

Exporting and Deleting Interval Data

You can export and delete interval data in batch mode using the INTDEXP.EXE command line program.

INTDEXP Command Syntax

INTDEXP uses the following syntax. Parameter switches are case insensitive; you can enter them in either upper or lower case (-c or -C). If a parameter includes a space, you must enclose it in quotes (for example, -s "11/01/1999 12:00:00").

```
intdexp -c connectstring [-q qualifier] [-f configfilename] [-i intdfilename] -o outfilename
[-s startdate] [-t stopdate] [-dtuse datetimerangeuse] [-d qualifiers [id list]]
[-ctrl controlfilename] [-m formatfilename] [-l logfilename] [-a archive] [-w export validation flag]
[-lcfg logging configuration filename]
```

In actual use, the command must be entered on one line. Also, you must either change the directory to the one where the program is stored (typically, \LODESTAR\bin) before entering the command, or specify the path in the command. To view a list of all parameters on-screen, type **intdexp -?** at the command prompt.

Parameter	Description
-c	<p><i>connectstring</i> is database connection information for the Oracle Utilities Data Repository. This parameter is required and must be in one of the following formats:</p> <p>For Oracle databases:</p> <pre>"Data Source=<data_source>;User ID=<user_id>;Password=<password>;LSProvider=ODP;"</pre> <p>For Microsoft SQL databases:</p> <pre>"Data Source=<address>;Initial Catalog=<SQL_database>;User Id=<user_id>;Password=<password>;MultipleActiveResultSets=True;LSP rovider=MSSQL;"</pre> <p>where:</p> <ul style="list-style-type: none"> <data_source> is the Oracle TNS Name for the data source, from the TNS_NAMES.ora file (typically located in the \\<machine>\oracle\network\admin directory) <user_id> is the user ID for the database connection <password> is the password for the supplied user ID. <address> is the IP address or Hostname of the MS SQL Server database server <SQL_database> is the name of the MS SQL Server database
-q	<i>qualifier</i> is an optional database qualifier. The default is PWRLINE.
-f	<i>configfilename</i> is the name of the configuration file that defines the working environment of the Oracle Utilities software (tells the software where to find and place the application data files, etc.). If you do not supply a value for <i>configfilename</i> , the system uses the default (LODESTAR.CFG). For information about the contents of this configuration file, see the <i>Oracle Utilities Energy Information Platform Configuration Guide</i> .

Parameter	Description
-i	<p><i>intdfilename</i> is the optional interval data source that you export the interval data from. You can specify an interval data file or interval data table in the Oracle Utilities Data Repository.</p> <ul style="list-style-type: none"> To specify a file, the path and file name of the interval data (.bte) file. To specify an interval data table, use the following format: RDB[/<table_name>] where: <table_name> is the optional name of the table from which the data is to be exported. Omitting the <table_name> indicates that the data should be exported from the table selected on the Interval Data Source tab on the General Options dialog <p>Note: INTDEXP does not support export from enhanced interval data tables.</p> <p>If not specified, the default is interval data source selected on the Interval Data Source tab on the General Options dialog (see Interval Data Source on page 2-12 in the <i>Data Manager User's Guide</i>).</p>
-o	<p><i>outfilename</i> is the name of the output file, in Oracle Utilities Standard Format (*.inp), Oracle Utilities Enhanced Format (*.lse), Oracle Utilities Standard XML Format (*.xml), or Oracle Utilities CSV Format (*.csv). This parameter is required.</p>
-s	<p>A start-date value is <i>optional</i>. You can use any of the following formats:</p> <p>Explicit Dates <i>mm/dd/yyyy hh:mm:ss</i> or <i>yyyy-mm-dd hh:mm:ss</i> Absolute value for date and time of the period's beginning. The time (hh:mm:ss) is optional, and defaults to midnight.</p> <p><i>mm/dd/yyyy-hh:mm</i> Alternate, acceptable format (used by Oracle Utilities Load Analysis).</p> <p>Relative Dates D## You can specify a relative date in days, where ## is the number of days before the current date and time, beginning at midnight (the "current date and time" are the date and time when the job starts). D0 represents the current date, beginning at midnight. D3 represents three days previous, beginning at midnight. M# You can specify a relative date in months, where # is the number of calendar months before the month containing the current date, starting at midnight on the first of the month. M0 is the current month, beginning on the 1st at midnight ("current date" refers to the date when the job starts).</p> <p>Note: mm, dd, and yyyy are all numeric. If you supply the time, you must enclose the parameter in double quotes, because the date and time are separated with a space; for example, "11/18/1998 01:00:00"</p>

Parameter	Description
-t	<p>A stop-time value is <i>optional</i>. You can use any of the following formats:</p> <p>Explicit Dates</p> <p><i>mm/dd/yyyy hh:mm:ss</i> or <i>yyyy-mm-dd hh:mm:ss</i></p> <p>Absolute value for date and time of the period's beginning. The time (hh:mm:ss) is optional, and defaults to midnight.</p> <p><i>mm/dd/yyyy-hh:mm</i></p> <p>Alternate, acceptable format (used by Oracle Utilities Load Analysis).</p> <p>Relative Dates</p> <p>D## You can specify a relative date in days, where ## is the number of days after the current date and time, beginning at midnight(the "current date and time" are the date and time when the job starts). D0 represents the end of current date, at 23:59:59. D3 represents three days after the current date, beginning at midnight.</p> <p>M# You can specify a relative date in months, where # is the number of calendar months after the month containing the current date, starting at midnight on the first of the month. M0 is the current month, beginning on the 1st at midnight ("current date" refers to the date when the job starts).</p> <p>Note: mm, dd, and yyyy are all numeric. If you supply the time, you must enclose the parameter in double quotes, because the date and time are separated with a space; for example, "11/18/1998 01:00:00"</p>
-dtuse	<p><i>datetimerangeuse</i> indicates how to use specified date-time range. Export cuts:</p> <ul style="list-style-type: none"> • TIMESTAMP (default) - whose TIMESTAMP falls within time range • STARTTIME - whose STARTTIME falls within time range • STOPTIME - whose STOPTIME falls within time range • WHOLE - that WHOLLY fall within time range • PARTIAL - that PARTIALLY fall within time range • SUBSET - PARTIAL plus SUBSETted to fit the time range
-d	<p><i>qualifiers [id/ list]</i> specifies the customer, account, recorder, or channel (either an ID or a list) to be used to limit exported cuts. The parameter is a 2-character code where the first character indicates whether it is a customer ('c'), account ('a'), recorder ('r'), or recorder,channel ('h'), and the second character indicates whether it is a list ('l'), ID ('i'), or all ('a'). If a list or ID is specified, the second parameter is followed by the list name or the ID.</p> <p>Note: If a list of recorders or channels is specified, it must be a table-column list of UIDRECORDERS or UIDCHANNELS.</p> <p>The names of the actual recorder/channels exported are retrieved from the Oracle Utilities Data Repository, based on these inputs. Recorders not in the relational database (such as *.BTE files) are not exported.</p>

Parameter	Description
-ctrl	<p><i>controlfilename</i> is an optional control file for specifying channels/cuts to export. The control file contains a list of channels and/or cuts to export. The format of this control file is:</p> <pre> <RECORDERID>,<CHANNELNUM> <RECORDERID>,<CHANNELNUM> ... </pre> <p>or</p> <pre> <RECORDERID>,<CHANNELNUM>,<STARTTIME> <RECORDERID>,<CHANNELNUM>,<STARTTIME> ... </pre> <p>where:</p> <ul style="list-style-type: none"> • <RECORDERID> is the Recorder ID for the channel/cut to export • <CHANNELNUM> is the Channel Number for channel/cut to export • <STARTTIME> is the Start Time of the cut to export <p>The first format is used to export all cuts for the specified channel(s). The second format is used to export only the specified cuts.</p> <p>Note: You can combine both formats in the same control file.</p> <p>Examples:</p> <p>Export all cuts for channel RECORDER_1,1:</p> <pre>RECORDER_1,1</pre> <p>Export the 01/01/2007 00:00:00 cut for RECORDER_1,1:</p> <pre>RECORDER_1,1, 01/01/2007 00:00:00</pre> <p>Note: This option is not supported when exporting data from a Pervasive.SQL (*.bte) file and using a Microsoft SQL Server database.</p>
-m	<p><i>formatfilename</i> is the optional format file for specifying additional attributes associated with the data to be exported. See Format File on page 8-39 for details about this file.</p>
-l	<p><i>logfile</i> is the name of an optional log file, used for output of error messages and additional information about the program's execution. If you do not use this parameter, the default log file, INTDEXPRPT, is placed in your USER directory.</p>
-a	<p><i>archive</i> is an optional archive switch to delete the cuts after they are exported.</p>

Parameter	Description
-w	<p><i>export validation flag</i> specifies how to set the Validate_Record_Flag on each cut in the export file. Can be one of the following:</p> <ul style="list-style-type: none"> v - Force Valid: Forces the Validate_Record_Flag on each exported cut to be set to “N”. i - Force Invalid: Forces the Validate_Record_Flag on each exported cut to be set to “Y” (Default). u - Use Flag: Uses the Internal_Validation flag and the Merge flag in the stored cut to determine how to set the Validate_Record_Flag when exporting. If either the Merge flag or the Internal_Validation flag are “Y”, then the Validate_Record_Flag in the exported cut is set to “N”. If both are set to “N” then the Validate_Record_Flag is set to “Y”.
-lcfg	<p><i>logging configuration filename</i> Name of an optional logging configuration file that specifies where error and log messages are sent. If you omit this parameter, the application creates a log file named INTDEXP.LOG in the LOG directory.</p>

Format File

The optional format file (-m) enables you to add descriptive data from the Oracle Utilities Data Repository to the header records of the exported cuts. Use the Format file to define the data you want to add, and how it should be organized. The data may come from any columns in the following Oracle Utilities Data Repository tables:

OPERATINGCOMPANY

CUSTOMER

ACCOUNT

CHANNELHISTORY

CHANNEL

RECORDER

PREMISE

RECORDERHISTORY

PHYSICALRECORDER.

The syntax for each line in the Format file is:

```
Header_Record_Sort_Code, TABLE_NAME.COLUMN_NAME[,
TABLE_NAME.COLUMN_NAME...]
```

Each line in the file represents a header record containing one or more attribute values to be added to the cut (in addition to the standard header records).

The first value is the sort code of the header record. Valid sort codes are from 00000100 to 09999999 (the leading zeros are optional in the Format file).

Note: This code is required to identify the type of record to the system. It does not affect the order in which the records are output. They will be output in the order in which you list them in the file.

After the sort code, you must supply one or more table.column indicators. Each of these indicators consists of a table name (from one of the eligible tables listed above) and a column name, separated by a period (.). The column name must be a valid column in the corresponding table. You must specify table and column names in all capital letters.

An example Format file is shown below.

```
00000100, OPERATINGCOMPANY.OPCOCODE  
00000101, RECORDER.UNINUMBER, PHYSICALRECORDER.METERNUMBER
```

This file defines two additional header records to be exported for each interval data cut. The first record uses sort code 00000100 and will contain the OPERATINGCOMPANY OPCOCODE column value. The second record uses sort code 00000101 and will contain the UNI Number (from the RECORDER Table) and meter number (from the PHYSICALRECORDER Table) attributes. Because the PHYSICALRECORDER table is related to the RECORDER table via a history table, the PHYSICALRECORDER record that was in effect at the end of the interval data cut's date range will be used.

Note: If the syntax of a specified format file is invalid, the INTDEXP program will fail and report the error.

Examples

The following commands will export all cuts from the default Btrieve database that have a TIMESTAMP of today's date:

```
INTDEXP -c mydsn -o outputfilename.lse  
-s "01/01/1999 00:00:00" -l mylogfile.txt
```

These commands will export all cuts from the default Btrieve database that have a STARTTIME of today's date:

```
INTDEXP -c mydsn -o outputfilename.lse  
-s "01/01/1999 00:00:00" -dtuse STARTTIME -l mylogfile.txt
```

Executing Oracle Utilities Rules Language Rate Schedules

This section explains how to set up RUNRS to apply the statements in a Rules Language rate form over a user-specified date range. This includes:

- **Purpose of RUNRS**
- **RUNRS Inputs**
- **RUNRS Command Syntax**

Purpose of RUNRS

The Run Rate Schedule (RUNRS) batch executable enables you to apply the power of the Oracle Utilities Rules Language without having to adhere to all of the strict billing rules required for Oracle Utilities Billing Component. You can use RUNRS to retrieve, manipulate, and write data to the Oracle Utilities databases; apply the general or special-purpose functions to any accessible data and perform calculations on that data; and format and report the data as desired.

To illustrate the purpose and potential of RUNRS, it is helpful to compare it to Oracle Utilities Billing Component (which is designed specifically to calculate bills). When any of the Oracle Utilities Billing Component programs run a rate form, they automatically apply the rate form calculations to the current account and bill period. You can use RUNRS command parameters to specify any desired date range, and you specify the data to be analyzed (if any) in the rate form.

RUNRS allows you to specify the analysis period as either an absolute date range, or some period relative to the day the job runs.

RUNRS Inputs

Before you can apply RUNRS, ensure that the following item is in place:

- **Rate Form:** The Rate Form must have an identifying record in the Data Repository, and must have been written in the Rules Language. If you are unfamiliar with this process, see the *Oracle Utilities Rules Language User's Guide* and the *Data Manager User's Guide*.

Note: When you construct or apply the rate form, keep in mind that RUNRS does not automatically supply a value for account-oriented database variables such as `account.accountid` and `billhistory.starttime`, as does Oracle Utilities Billing Component.

Important: When you run a rate schedule without specifying an account, certain functions (e.g., `INTDLOADHIST` and `INTDLOADUOM`) may not work properly.

RUNRS Command Syntax

The command to run RUNRS uses the following syntax. Parameter switches are case insensitive; you can enter them in either upper or lower case (-c or -C). If a parameter includes a space, you must enclose it in quotes (for example, -s "11/01/1999 12:00:00").

```
runrs -c connectstring [-q qualifier] [-f configfilename] -v OPCO:JUR:RS[:VER]
[-s startdate] [-t stopdate] [-a (a|c)(a|i|l[*]|f) name] [-x xmlfilename] [-b intdb]
[-w outputname] [-e [#]] [-k saveresults] [-i# intervaldatahandling] [-rp reportoptions]
[-rop reportoptions] [-lcfg logging configuration filename]
```

In actual use, the command must be entered on one line. Also, you must either change the directory to the one where the program is stored (typically, \LODESTAR\bin) before entering the command, or specify the path in the command. To view a list of all parameters on-screen, type **runrs -?** at the command prompt.

Parameter	Description
-c	<p><i>connectstring</i> is database connection information for the Oracle Utilities Data Repository. This parameter is required and must be in one of the following formats:</p> <p>For Oracle databases:</p> <p>"Data Source=<data_source>;User ID=<user_id>;Password=<password>;LSPProvider=ODP;"</p> <p>For Microsoft SQL databases:</p> <p>"Data Source=<address>;Initial Catalog=<SQL_database>;User Id=<user_id>;Password=<password>;MultipleActiveResultSets=True;LSPProvider=MSSQL;"</p> <p>where:</p> <ul style="list-style-type: none"> • <data_source> is the Oracle TNS Name for the data source, from the TNS_NAMES.ora file (typically located in the \\<machine>\oracle\network\admin directory) • <user_id> is the user ID for the database connection • <password> is the password for the supplied user ID. • <address> is the IP address or Hostname of the MS SQL Server database server • <SQL_database> is the name of the MS SQL Server database
-q	<i>qualifier</i> is an optional database qualifier. The default is PWRLINE.
-f	<p><i>configfilename</i> is the name of the configuration file that defines the working environment of the Oracle Utilities software (directs the software where to find and place the application data files, and so on). If you do not supply a value for <i>configfilename</i>, the system uses the default (LODESTAR.CFG). For information about the contents of this configuration file, see the <i>Oracle Utilities Energy Information Platform Configuration Guide</i>.</p>

Parameter	Description
-s	<p>A start-date value is required. You can use any of the following formats:</p> <p>Explicit Dates <code>"mm/dd/yyyy hh:mm:ss"</code> or <code>"yyyy-mm-dd hh:mm:ss"</code> Absolute value for date and time of the period's beginning. Time is optional; midnight is the default.</p> <p><code>"mm/dd/yyyy hh:mm"</code> Alternate, acceptable format (used by Oracle Utilities Load Analysis). If this format is used, times are adjusted. A start time of 00:01 will automatically be changed to 00:00:00, and a stop time of 24:00 will become 23:59:59.</p> <p>Relative Dates D## You can specify a relative date in days, where ## is the number of days before the current date and time, beginning at midnight (the "current date and time" are the date and time that the RUNRS job starts). D0 represents the current date, beginning at midnight. D3 represents three days previous, beginning at midnight.</p> <p>M# You can specify a relative date in months, where # is the number of calendar months before the month containing the current date, starting at midnight on the first of the month. M0 is the current month, beginning on the 1st at midnight. (Again, "current date" refers to the date when the RUNRS job starts.)</p> <p>Note: mm, dd, and yyyy are all numeric. If you supply the time, you must enclose the parameter in double-quotes, because the date and time are separated with a space; for example, "11/18/1998 01:00:00"</p>

Parameter	Description
-t	<p>A stoptime value is <i>optional</i>. You can use any of the following formats:</p> <p>Explicit Dates <code>mm/dd/yyyy hh:mm:ss</code> or <code>yyyy-mm-dd hh:mm:ss</code> Absolute value for date and time of the period's beginning. Time is optional; midnight is the default.</p> <p><code>mm/dd/yyyy-hh:mm</code> Alternate, acceptable format (used by Oracle Utilities Load Analysis). If this format is used, times are adjusted. A start time of 00:01 will automatically be changed to 00:00:00, and a stop time of 24:00 will become 23:59:59.</p> <p>Relative Dates D## You can specify a relative date in days, where ## is the number of days after the current date and time, beginning at midnight (the "current date and time" are the date and time that the RUNRS job starts). D0 represents the current date, at 23:59:59. D3 represents three days previous, beginning at midnight.</p> <p>M# You can specify a relative date in months, where # is the number of calendar months after the month containing the current date, starting at midnight on the first of the month. M0 is the current month, beginning on the 1st at midnight. (Again, "current date" refers to the date when the RUNRS job starts.)</p> <p>Note: mm, dd, and yyyy are all numeric. If you supply the time, you must enclose the parameter in double-quotes, because the date and time are separated with a space; for example, "11/18/1998 01:00:00"</p>
-v	<p><code>OPCO:JUR:RS[:VER]</code> is the identifier for the rate form. It is required (except when you supply an account ID/list using the -a switch, and corresponding records exist in the Rate Code History table). OPCO, JUR, RS are the operating company code, jurisdiction code, and rate form code (respectively) that identify the rate form.</p> <p>You can specify a NULL operating company or jurisdiction by leaving a blank space where OPCO or JUR would be.</p> <p>The version number is optional. If you do not supply a version number, the program automatically uses the rate form that was applicable on the stop date of the analysis period (specified using the -t switch). If you supply a version number, the program uses the trial version indicated (you would typically do this for testing). See the <i>Data Manager User's Guide</i> for additional information about this convention for identifying rate forms.</p>

Parameter	Description
-a	<p><i>(a c)(a i l[*] f) name</i> Optional Account/Customer All/Id/List/File name; default is no account. An * immediately after l means refresh the list.</p> <p>If -aai is specified, you can save to the Bill History Table.</p> <p>If -aaf is specified, the file must be one of the following:</p> <ul style="list-style-type: none"> a list of customer/account IDs (*.ids) a list file (*.lst) an account query file (*.qra) a customer query file (*.qrc). <p>Examples:</p> <ul style="list-style-type: none"> All Accounts: -aaa Account ID: -aai555888 Account List: -aalMAC_RUNRS_ACCTLIST Account ID File: -aafMAC_RUNRS_ACCTFILE.ids. This file exists in the specified user directory (defined on the User Files tab of the Default Options dialog) <p>Note: The Stop Time parameter is required if you supply this parameter.</p>
-ax	<p>-ax indicates not to use the account selection to set up the run, but to pass it to the rate schedule. This must appear after the <i>-a(a c)(a i l[*] f) name</i> option. An * immediately after l means refresh the list.</p>
-x	<p>Either a <i>filename</i> or <i>id.type=value</i>.</p> <p><i>filename</i> Input values for rate schedule identifiers. The file must be in XML format (see XML File Format, below). The file name must include the full path.</p> <p><i>id.type=value</i> Set identifier id to this value. Identifier id (name) must be uppercase. “.type” is optional and may be f (float), i (integer), d (datetime), or s (string). The default is string (s).</p> <p>The -x parameter may be repeated with different ids and files. The two formats (file and id) may be intermixed. Elements in multiple .xml files will be concatenated. Duplicate assignments to the same identifier may generate an error if the types differ.</p>
-b	<p><i>filename</i> is the name of the Interval Database that contains the interval data for the calculations. The default is selected via the Interval Data Source tab on the General Options dialog.</p> <p>A file name that is simply RDB specifies the interval data table selected on the Interval Data Sources tab on the Default Options Data Manager dialog (see Interval Data Source on page 2-12 in the <i>Data Manager User's Guide</i>).</p>
-w	<p><i>filename</i> is the name that the program assigns to the output reports. If you do not supply this option, the program displays the output on screen and does not produce a file.</p>

Parameter	Description
-e	tells the program whether to write errors to an output file. If you use this switch, you must also supply a value for -w to designate a name for the file (otherwise, the program will not create the file). If you omit the -e switch, the program displays error messages on-screen only. If you specify the -e switch with the optional argument (any number greater than 1), the output report contains status messages as well as errors (e.g., -e2).
-k	<i>saveresults</i> tells the program to save the results, as specified in the rate form. If you omit this switch from the command, the program ignores all SAVE statements in the rate form (this is useful if you want to debug the rate form before using it in a production mode). If you supply the switch, the program saves interval data, table saves, and CIS records. Because RUNRS does not input an account, determinants are not saved to a Bill History record, unless -ai is specified (see above). Note: To export/archive any records overwritten by the new SAVED records, in Data Manager select Tools->Options->Interval Data Source and check the box next to Export Records Overwritten by SAVE to File . In the text box below, supply the full path and filename for the file the program will export to.
-i#	Set interval data handling. Same values as INTD_ERROR_STOP (0,1,2,3). Default # is 1 (error stop). If not set, the On Rules Language Interval Data Error option specified on the Error Handling tab of the General Options dialog is used.
-rp	Turn on all Report Options - Problem Determination.
-rop	<i>eibr</i> Any combination of e i b r. Sets Report Options - Problem Determination. Turn on Error Messages, Input options, Show Bill History, Show Rate Schedule.
-lcfg	<i>logging configuration filename</i> Name of an optional logging configuration file that specifies where error and log messages are sent. If you omit this parameter, the application creates a log file named RUNRS.LOG in the LOG directory.

Example

A sample RUNRS command is:

```
RUNRS -cpowrln -sD41 -tD2 -vGECO:SF:MRSP -k
```

At the conclusion of the job run, RUNRS returns a 0 if successful, and 99 if any error occurred (including initialization or validation errors, as well as any errors that occurred while running the rate schedule).

To run a rate form with a NULL operating company or jurisdiction, leave a blank where the OPCO or JUR code would be in the command line.

```
RUNRS -cpowrln -sD41 -tD2 -v::RATE -k
```

-X Parameter

The -X parameter allows identifier values to be passed to the rate schedule in one of two formats:

- **XML File format**
- **ID Values**

XML File format

The xml file (used with the -x switch) must consist of one root element - RATE_SCHEDULE_INPUT_PARMS.

This element then contains one or more sub elements.

The format of a sub element is:

```
<IDENTIFIER TYPE="F|I|D|S">value</IDENTIFIER>
```

Identifier names must be uppercase. TYPE is optional (s is the default). The types are:

F = float, I = integer, D = datetime, S = string.

Leading and trailing blanks are removed from the value.

Example:

The following XML file would pass the following values to the rate schedule:

- TEST_DATE = 2000-06-30
- TEST_KWH = 900.0

```
<RATE_SCHEDULE_INPUT_PARMS>
  <TEST_DATE TYPE="D">2000-06-30</TEST_DATE>
  <TEST_KWH TYPE="F">900.0</TEST_KWH>
</RATE_SCHEDULE_INPUT_PARMS>
```

ID Values

Sets identifiers to supplied values in the following format:

```
-x<ID>.<TYPE>=<VALUE>
```

where:

- **<ID>** is the Rules Language Identifier to which the value will be passed (must be uppercase)
- **<TYPE>** optional type of identifier. May be F (float), I (integer), D (datetime), or S (string). The default is string (S).
- **<VALUE>** is the value to be passed to the identifier.

Example:

The following -x parameters would pass the following values to the rate schedule:

- TEST_DATE = 2000-06-30
- TEST_KWH = 900.0

```
"-xTEST_DATE.D=2000-06-30" "-xTEST_KWH.F=900.0"
```

Automated Profiling

This section explains how to use the Automated Profiler (AUTOPROFEXE) to estimate a load profile for an account by “spreading” its metered kWh value over a time period according to a class profile or other template. This includes:

- **Purpose of AUTOPROF**
- **AUTOPROF Inputs and Other Prerequisites**
- **AUTOPROF Command Syntax**
- **AUTOPROF Processing**

Purpose of AUTOPROF

The Automated Profiler (AUTOPROF) estimates the load shape for an account using the account’s metered kWh and a generic class template.

You define the profiling calculations in a Rules Language rate form (see **Example of a Profiler Rate Form** on page 8-49). The rate form uses the template to determine the percentage of the account's total energy consumed relative to the class for each interval, thereby “spreading” the account’s total energy for the period over the same load shape as the template.

When the job is submitted, the Automated Profiler retrieves the account’s kWh for the bill period, the start- and stop-times for the bill period, and the template. It performs the calculations and saves the resulting interval data according to the instructions in the rate form.

AUTOPROF Inputs and Other Prerequisites

Prerequisites to running the Auto Profiler include:

- **Account Bill History Profile Status Flag:** This flag has three possible values: **Null** (do not profile), **N** (not yet profiled, to be profiled), and **Y** (profiled). The Automated Profiler looks for Bill History records whose value is N. When the Automated Profiler successfully profiles this account for its bill period, it resets the flag to Y. If you change the profile (either by changing the kWh value or by changing the generic template), you can reset the flag to N (in which case the Automated Profiler will create a new profile during its next run).

For every account whose profile will be estimated by the Automated Profiler, you must set the Profile Status Flag to N in the account’s Bill History record in the Data Repository.

You can set the flags using the Import utility or the Data Manager Browser; see the *Data Manager User’s Guide* for instructions.

- **Template:** The generic profile must cover at least the same period as the bill history record for the account to be profiled (defined by the start- and stop-dates in the account's bill history record for the current period). Oracle Utilities recommends that the frequency of the intervals in the template be the same as that desired for the output profile.
- **Profiler Rate Form:** The Automated Profiler requires you to supply a Rules Language rate form. This rate form tells the Automated Profiler where to find the template, how to spread the kWh value proportionately to each interval in the period, and how to set the status codes in the resulting interval data cut to indicate that it has been profiled. You should also incorporate routines that save the profiled cut to the Interval Database, then read it back in, verify it, and ABORT if an error occurs. If you are unfamiliar with the Rules Language and how it is used to construct rate forms, see the *Oracle Utilities Rules Language User’s Guide*.

Example of a Profiler Rate Form

```

/* Account's generic profile recorder is its revenue code, channel one
*/
GENERIC_PROFILE_RCDCHAN = ACCOUNT.REVENUECODE + ",1";
/* This load will automatically use the dates from the BILLHISTORY
record */
GENERIC_PROFILE_HNDL = INTDLOAD(GENERIC_PROFILE_RCDCHAN);
/* Spread the KWH value proportionately to each interval*/
PROFILE_HNDL = GENERIC_PROFILE_HNDL * (KWH /
GENERIC_PROFILE_HNDL.TOTAL);
/* Set the status codes*/
SC = IDATTR(KWH, "STATUSCODE");
REPORT SC LABEL "kWh Status Code";
PROFILE_HNDL.STATUSCODE = SC;
/* The account profile's recorder id is the account id */
PROFILE_HNDL.RECORDER = ACCOUNT.ACCOUNTID;
PROFILE_HNDL.CHANNEL = 1;
DESC = "Generic Profile " + ACCOUNT.REVENUECODE + " for " +
ACCOUNT.ACCOUNTID;
REPORT DESC LABEL "Profiled cut descriptor";
PROFILE_HNDL.DESRIPTOR = DESC;
/* Set origin to profiled */
PROFILE_HNDL.ORIGIN = "P";
/* Save the profiled cut to the interval data database */
SAVE PROFILE_HNDL;
/* Validate saved cut */
TEST_RCDCHAN = ACCOUNT.ACCOUNTID + "," + 1;
/* This load will automatically use the dates from the BILLHISTORY
record */
TEST_HNDL = INTDLOAD(TEST_RCDCHAN);
IF TEST_HNDL = 0
    THEN
        ABORT "Error loading profile cut " + TEST_RCDCHAN;
END IF;
/* Compare original to read in */
IF INTDISEQUAL(PROFILE_HNDL, TEST_HNDL) = 0
    THEN

```

AUTOPROF Command Syntax

The command to run AUTOPROF uses the following syntax. Parameter switches are case insensitive; you can enter them in either upper or lower case (-c or -C). If a parameter includes a space, you must enclose it in quotes.

```
autoprof -cconnectstring [-q qualifier] -fconfigfilename -vOPCO:JUR:RS[:VER]
-bintdb -d billdeterminants -e[#] erroroutput -woutputname -k saveresults
[-lcfg logging configuration filename]
```

In actual use, the command must be entered on one line. Also, you must either change the directory to the one where the program is stored (typically, \LODESTAR\bin) before entering the command, or specify the path in the command. To view a list of all parameters on-screen, type **autoprof -?** at the command prompt.

Parameter	Description
-c	<p><i>connectstring</i> is database connection information for the Oracle Utilities Data Repository. This parameter is required and must be in one of the following formats:</p> <p>For Oracle databases:</p> <pre>"Data Source=<data_source>;User ID=<user_id>;Password=<password>;LSPProvider=ODP;"</pre> <p>For Microsoft SQL databases:</p> <pre>"Data Source=<address>;Initial Catalog=<SQL_database>;User Id=<user_id>;Password=<password>;MultipleActiveResultSets=True ;LSPProvider=MSSQL;"</pre> <p>where:</p> <ul style="list-style-type: none"> • <data_source> is the Oracle TNS Name for the data source, from the TNS_NAMES.ora file (typically located in the \\<machine>\oracle\network\admin directory) • <user_id> is the user ID for the database connection • <password> is the password for the supplied user ID. • <address> is the IP address or Hostname of the MS SQL Server database server • <SQL_database> is the name of the MS SQL Server database
-q	<i>qualifier</i> is an optional database qualifier. The default is PWRLINE.
-f	<p><i>configfilename</i> is the name of the configuration file that defines the working environment of the Oracle Utilities software (directs the software where to find and place the application data files, and so on). If you do not supply a value for <i>configfilename</i>, the system uses the default (LODESTAR.CFG). For information about the contents of this configuration file, see the <i>Oracle Utilities Energy Information Platform Configuration Guide</i>.</p>

Parameter	Description
-v	<i>OPCO:JUR:RS[:VER]</i> is the identifier for rate form (see previous page). It is required . OPCO, JUR, RS are the operating company code, jurisdiction code, and rate form code that identify the rate form. The version number is optional. If you do not supply a version number, autoprof automatically uses the rate form that was applicable on the account's bill history stop date. If you supply a version number, autoprof uses the trial version indicated (you would typically do this for testing). See the <i>Data Manager User's Guide</i> for additional information about this convention for identifying rate forms.
-b	<p><i>intdb</i> is the full path and name of the Interval Database that contains the profile template. The default is selected via the Interval Data Source tab on the General Options dialog.</p> <p>A file name that is simply RDB (-bRDB) specifies the interval data table selected on the Interval Data Sources tab on the Default Options Data Manager dialog (see Interval Data Source on page 2-12 in the <i>Data Manager User's Guide</i>).</p>
-d	<i>billdeterminants</i> specifies a comma-separated list of bill determinant identifiers (from the IDENTIFIER column of the Bill Determinants Table). All determinants listed are retrieved for each account and their values passed to the specified rate form. The determinants listed must be stored in the Bill History Table (not the Bill History Value Table). If this option is not supplied, kWh (the default) is used. The maximum length for this parameter is 256 characters.
-e	<i>erroroutput</i> tells autoprof whether to write errors to an output file. If you use this switch, you must also supply a value for -w to designate a name for the file (otherwise, autoprof will not create the file). If you do not specify the -e switch, the program displays error messages on-screen only. If you specify the -e switch with the optional argument (any number greater than 1), the logfile contains status messages as well as errors (e.g., -e2).
-w	<i>outputname</i> is the name autoprof assigns to the output reports. If you do not supply this option, the program displays the output on-screen and does not produce a file.
-k	<i>saveresults</i> tells autoprof to save the results, as specified in rate form. If you omit this switch from the command, autoprof ignores all SAVE statements in the rate form (this is useful if you want to debug the rate form before using it in a production mode, for example). If you supply the switch, autoprof writes the results to the Interval Database (as identified by the -b parameter).
-lcfg	<i>logging configuration filename</i> Name of an optional logging configuration file that specifies where error and log messages are sent. If you omit this parameter, the application creates a log file named AUTOPROF.LOG in the LOG directory.

Note: A setting in the Data Manager Options allows you to export/archive any records overwritten by the new SAVED records. In Data Manager, select **Tools->Options**, and select the **Interval Data Source** tab. Next, check the **Export Records Overwritten by SAVE to File** box. In the text box below that option, supply the full path and filename for the file to which the program will export.

Example

A sample AUTOPROF command is:

```
AUTOPROF -capuobent -vGECO:MA:S1 -k
```

AUTOPROF Processing

The Automated Profiler gets a list of all Bill History records in the Data Repository with the profile Status Flag set to N. It applies the calculations in the Profiler rate form to the account. Specifically, the Profiler gets the account, and its bill period start and stop dates. It runs the profiler rate form using the account and the account's start and stop dates as its parameters. When the calculations are complete, the rate form saves the new cut (profile) to the Interval Database, setting the Origin flag in the cut's header record to P to indicate that the cut was produced by the Profiler.

Finally, if the rate schedule completed successfully, the Profiler updates the Profile Status Flag in the account's Bill History record for the current bill period to Y; otherwise, it leaves the flag set to N and writes an EXCEPTION note to the ACCOUNTNOTE Table for the account.

Migrating RCDATA

This section explains how to use RCTORDBM to move rate forms, lists, and queries from the RCDATA directory used in older versions of Oracle Utilities software into tables in the Oracle Utilities Data Repository. This includes:

- **Purpose of RCTORDBM**
- **RCTORDBM Command Syntax**

Purpose of RCTORDBM

In previous versions of the Oracle Utilities Energy Information Platform (including Oracle Utilities Billing Component, Oracle Utilities Rate Management, and Oracle Utilities Load Profiling and Settlement), the rate forms, lists, and queries (as well as Revenue Summary and Bill Frequency data) used by the Oracle Utilities analysis programs were stored in the RCDATA directory, a shared directory on the application server.

New versions of Oracle Utilities products store this data in relational tables in the Oracle Utilities Data Repository.

The RCDATA to Relational Database (RCTORDBM) batch executable enables you to move rate forms, lists, and queries from the RCDATA directory into tables in the Oracle Utilities Data Repository.

In most implementations, you'll only have to use RCTORDBM once, to move your existing rate forms, list, queries, etc. into the Oracle Utilities Data Repository. However, if you need to import older data into the RCDATA directory, you can use this program to move that data into the database.

RCTORDBM Command Syntax

The command to run RCTORDBM uses the following syntax. Parameter switches are case insensitive; you can enter them in either upper or lower case (-c or -C). If a parameter includes a space, you must enclose it in quotes (for example, -s "11/01/1999 12:00:00").

rctordbm -c *connectstring* [-q *qualifier*] -f *configfilename* -r *rateformdata*
 -v *revenuesummarydata* -b *billfrequencydata* -s *queryandlistdata* -a *alldata* -l *logfile* -e *error*

In actual use, the command must be entered on one line. Also, you must either change the directory to the one where the program is stored (typically, \LODESTAR\bin) before entering the command, or specify the path in the command. To view a list of all parameters on-screen, type **rctordbm -?** at the command prompt.

Parameter	Description
-c	<p><i>connectstring</i> is database connection information for the Oracle Utilities Data Repository. This parameter is required and must be in one of the following formats:</p> <p>For Oracle databases:</p> <p>"Data Source=<data_source>;User ID=<user_id>;Password=<password>;LSPProvider=ODP;"</p> <p>For Microsoft SQL databases:</p> <p>"Data Source=<address>;Initial Catalog=<SQL_database>;User Id=<user_id>;Password=<password>;MultipleActiveResultSets=True;LSPProvider=MSSQL;"</p> <p>where:</p> <ul style="list-style-type: none"> • <data_source> is the Oracle TNS Name for the data source, from the TNS_NAMES.ora file (typically located in the \\<machine>\oracle\network\admin directory) • <user_id> is the user ID for the database connection • <password> is the password for the supplied user ID. • <address> is the IP address or Hostname of the MS SQL Server database server • <SQL_database> is the name of the MS SQL Server database
-q	<i>qualifier</i> is an optional database qualifier. The default is PWRLINE.
-f	<p><i>configfilename</i> is the name of the configuration file that defines the working environment of the Oracle Utilities software (directs the software where to find and place the application data files, and so on). If you do not supply a value for <i>configfilename</i>, the system uses the default (LODESTAR.CFG). For information about the contents of this configuration file, see the <i>Oracle Utilities Energy Information Platform Configuration Guide</i>.</p>
-r	<i>rateformdata</i> Include this switch to convert rate form data (*.rlf) from RCDATA to the Data Repository.
-v	<i>revenuesummarydata</i> Include this switch to convert Revenue Summary data (*.rev) in RCDATA to the Data Repository.

Parameter	Description
-b	<i>billfrequencydata</i> Include this switch to convert Bill Frequency data (*.bft) in RCDATA to the Data Repository.
-s	<i>queryandlistdata</i> Include this switch to convert Query and List data (*.qry and *.lis) in RCDATA to the Data Repository.
-a	<i>alldata</i> Include this switch to convert All data in RCDATA to the Data Repository.
-l	<i>logfile</i> is the name of the optional logfile used for output of error messages and additional information about the program's execution. If you omit this parameter, it defaults to RCTORDBM.RPT, which is placed in your USER directory.
-e	<i>error</i> Include this switch if you want only error messages to be printed to the logfile. If you supply this switch, no status messages are included in the logfile.

Note: At least one of the following switches is required: -r, -v, -b, -s, -a.

Example

A sample RCTORDBM command is:

```
RCTORDBM.EXE -clodestar -fc:\lodestar.cfg
-a -lc:\loadstar\user\error.log
```

This converts all RCDATA into the Relational database.

Chapter 9

Creating a CIS Transaction Record Output File

The SAVE x to CIS statements in a rate form write selected results of billing or settlement calculations to an output file, which is in turn read by a Customer Information System (CIS) and/or other company system. The Transaction File, described in this chapter, tells the programs how to format the contents of this output file so that it is readable by the CIS or other system. The Transaction File must include separate formatting instructions for each CIS transaction type that is used at your company; for example, simple bills, complex bills, cancel/rebills.

Note: The amount of information that Oracle Utilities Billing Component writes to the output file is also affected by the **Print Detail Option** setting (**Default**, **None**, **Normal**, or **All**).

Transaction Type Description File Format

This chapter explains how to set up the Transaction File. This file is required if you wish to output the results of a bill or settlement calculation, and then transfer those results to your CIS and/or other company system.

The Transaction File Format provides you the flexibility to output this information with the Oracle Utilities Rules Language and a user-defined custom configuration output format.

After the bill or settlement information is calculated in Oracle Utilities Billing Component or in Oracle Utilities Load Profiling and Settlement, the results can be formatted and written to an output file for your CIS to process.

There are two primary files needed in order to generate and to transfer your results. One is an Output File and the other is a Control File.

Output File

You must specify the name of the output file in the LODESTAR.CFG file, an entry — CISFILENAME = D:\USER\LODESTAR.CIS — to define where the Oracle Utilities application will place the results generated by the defined fields in the control file and Rate Schedule that will be transferred to your CIS system. The user must define and specify full path for this entry in the LODESTAR.CFG file.

Note: The name of the output file is user-defined. If left undefined, the default is LODESTAR.CIS.

You can also output results to a specific CIS file by using the Optional File Name in the SAVE TO CIS statement (see the *Oracle Utilities Rules Language User's Guide*). SAVE TO CIS statements that do not specify a File Name will write records to the default CIS file (specified in the LODESTAR.CFG file).

Important Note

Detail record types (31, 41, 51, and 61) can only be saved to the default CIS file, and **CANNOT** be saved to specific CIS files. See **Record Types** for more information.

Control File

You must create a Transaction File, that is a **control file** that identifies any information that you may wish to transfer to your CIS or other company system. The default name of this file is "CISFORMAT.TXT". By default the name of this file should be defined in the LODESTAR\CFG directory. The control file is an ASCII text file, that contains a description of each CIS transaction type.

To designate a control file, include the line CISFORMAT = (*path and filename*) in the configuration file. If unspecified, the system will look for a file named CISFORMAT.TXT in the LODESTAR\CFG directory. In the LODESTAR.CFG file, an entry can be added to define where the software can locate the control file (for example, "CISFORMAT = D:\USER\APPB.CTR"). To specify a control file at the rate schedule level (a file that will be used for the currently executing rate schedule), using the LSRSENV.CISFORMAT_FILENAME parameter (for example, "LSRSENV.CISFORMAT_FILENAME = ALTCTL.TXT").

You must define each transaction type within a Rate Schedule, and the CIS transactions are generated via the **SAVE x TO CIS** Rules Language statement within the schedule (see the *Oracle Utilities Rules Language User's Guide*).

Note: The following applies only when x is a stem identifier that was referenced earlier in the rate form. If x is a simple identifier with a string value, the string is written to the CIS file exactly as is, with no formatting, leading or trailing separators, etc.

In addition, the SAVE TO CIS statement provides the capability to specify an Optional Section Name that has been defined in the CISFORMAT.TXT file. (For more information on the SAVE TO CIS statement, see the *Oracle Utilities Rules Language User's Guide*.)

For each transaction type, the control file defines all the Field Labels and their formats. The formats used in the file are Standard COBOL format definitions. The control is an ASCII text file that the user can modify in the future, if needed, using any text editor. Blank/empty lines will be allowed anywhere in the file to improve the readability of the file. If a line starts with a semi-colon (";") the line is ignored - it can be used as a comment.

This control file is very flexible and allows you to define a variable number of sections for different output formats (CIS transaction type).

In the first section of the control file, you can define control keywords that affect the format of the output file. For example, you can specify the number of sections, the number of identifiers per section, and the header by these control keywords.

The control keywords are:

Keyword	Meaning
[NOLEADINGZERO]	Do not display leading zeros in numeric fields.
[NOLABEL]	Do not display the identifier names in the output file.
[NODETAIL]	Do not include detail records in the output file.
[DETAILLINECT]	##, where ## is the number of lines per CIS record. The default is 70.
[NOTRAILSEP]	Do not include a trailing separator in each record.
[SEPARATOR];,,	Use the character between the semicolons as the column separator. To specify a non-printing character, such as a Tab or Carriage Return, use the decimal ASCII code value for the character with the following format: single quote - escape (“\”) - ASCII code - single quote Examples: Tab - [SEPARATOR];'\9'; Carriage Return - [SEPARATOR];'\13';
[NOSEPARATOR]	Do not use a separator at all. This requires that all fields have fixed width. It overrides [SEPARATOR] if it comes second, otherwise the [SEPARATOR] value will be used.
[INITIALIZE]	Initialize missing values for numeric fields to zero, and missing string values to a single space (the actual field will be filled according to its format). Otherwise, if the width is fixed it is filled with spaces, if it is not fixed (there is a separator) the field is empty.
[INVOICENUMBER]	Indicates that approved bills require invoice numbers.
[STRICTTYPECHECK]	Verify that the identifier has the same type as the format string. If this is not used the identifier value is converted to match the format type, then the value is formatted. If this is used it is an error for a user assigned value to be a different type from its corresponding format field. Note that PIC 9 formats correspond to Integer and Float types, and PIC X formats correspond to String and Date types. All other value types will always generate an error. It is always an error for a system assigned value to differ from its format. The system assigned values are listed below.
[HEADER];##	## is the number of identifiers/columns in the common header part of each record (optional).
[SECTION];##	## is the number of identifiers/columns in the transaction specific part of each record.
=====	End this part of the control file.

Entries in the header and sections consist of Field Label-COBOL format pairs, separated by semicolon (;). There will be one entry per line with each entry containing two columns: Field label and COBOL format. The field label is the name of the Rules Language identifier that contains the column's value. The format of the value to be written to the output file is defined using one of the following standard COBOL picture formats:

Format	Meaning
PIC X	any number of characters
PIC X(1)	one character
PIC XXX	three characters
PIC X(3)	three characters
PIC 9	one or more numbers (integer only)
PIC 9.99	one or more numbers left of decimal, the decimal point, and two digits right of the decimal (invalid if NOSEPARATOR is set)
PIC 9(1)	one number
PIC 999	three numbers
PIC 9(3)	three numbers
PIC 9(8).99	eight digits to the left of the decimal, the decimal point, and two digits to the right of the decimal
PIC 9(8).99-	the same with a trailing sign (only displayed if negative, invalid if NOSEPARATOR is set)
PIC -9(8).99	the same with a leading sign (only displayed if negative, invalid if NOSEPARATOR is set)
PIC 9(8)V99	eight digits to the left of the decimal, two digits to the right of the decimal, but no actual decimal point (field is 10 character wide)
PIC 9(8)V99S	the same with a trailing sign (+ or - displayed)
PIC S9(8)V99	the same with a leading sign (+ or - displayed).

Anything else is copied as is (no value substitution) after leading and trailing spaces are removed.

There are three types of sign characters that you can specify within the format, - '+' (plus), '-' (minus) and 'S'. '+' '-' are the same and cause a minus sign to appear if the value is negative, otherwise no character is output. 'S' causes a minus sign to appear if the value is negative, otherwise a plus sign is output. Only one should be used per line, it can be either leading or trailing. If NOSEPARATOR is used only S is valid, since the width is fixed.

Also, there are two decimal characters that you can specify within the format, - '.' (period) and 'V'. '.' causes a period to appear in the decimal position. 'V' outputs only the numeric characters, without a decimal point.

Note: If the width specified before the decimal point (the value in the parentheses) is insufficient to hold the value, the full value will be output (no left truncation). Hence fixed width fields must allow enough width to display any value for the field.

The use of [NOSEPARATOR] implies fixed width fields. Every field must have an explicit length - X(##) or 9(##)... - and only 'S' may be used as a sign character. Missing values will be blank filled. If the value is a number and it is too big for the field - number is 12345 and field is PIC 9(4), for example - the field is filled in with number signs (##### in this example).

For system assigned values, and user assigned values if `STRICTTYPECHECK` is used, if the format specifies a string and the corresponding identifier is not a string, the following string will be placed in the CIS file:

Error: PIC Xs value ##.##.

If the format specifies a number and the corresponding identifier is not a number, the following string will be placed in the CIS file:

Error: PIC 9s value 'XXX'.

The system-assigned identifiers that cannot be overridden by the user are:

`INTFC_ACC_ID`, `CUST_NAME`, `INTFC_CUST_ID`, `TRANS_EFF_DATE`, and `INTFC_RCD_TY`.

The system-assigned identifiers that can be overridden by the user are:

`BILL_HIST_PERIOD`, `READ_FROM_DT`, `READ_TO_DT`, `CIS_BILL_TYPE`, `TURN_OFF_DT`, `REV_MONTH`, and `INVOICE_NUMBER`.

Note that the format of system assigned dates is always the string format "YYYY-MM-DD".

Hyphens (-) are not allowed for definition of identifiers. If there is a value for an identifier in the Rate Form, the Field Label and value for that identifier will be written to the output file. For example, "STATE_TAX_AMT;500;". If there is no value associated with the identifier in the Rate Form, only the delimiter is output. For example, "STATE_TAX_AMT;". The Field Label is only displayed if `[NOLABEL]` is not on. The ; can be changed to another character using the `[SEPARATOR]` keyword.

Only the Field Labels listed in the control file will be used to generate the correct output format to CIS. Any identifier in the Rate Form that does not have a corresponding Field Label will not be written to the output file.

The special label `FILLER` can be used to put spaces in the record. Its COBOL format should always be `X(##). ##` spaces will be placed in the output record. If `[NOLABEL]` is not set, "FILLER" will be used as the label in the record.

The header section begins with the keyword "`[HEADER]`" and ends with key string "=====" (ten = signs). Each line in it has a number of Field Label-COBOL format pairs.

Each section begins with the keyword "`[SECTION]`" and ends with key string "=====" (ten = signs). A section may have a name. If so, the name appears after the `[SECTION]` delimiter, on the same line. There must be one section with the name `CANCEL` if you want to do Cancel or Cancel/Rebill processing in Bill Correction.

If an account requires an EDI transaction (EDI column in `ACCOUNT` table record set to Y), and you want a separate format for the EDI transaction, do the following: Create a section, with a name, that defines the regular transaction format. Then create another section whose name is the same, but with "`_EDI`" appended. Put the EDI transaction formats in this section. Oracle Utilities Billing Component will automatically use the correct section for the two transaction types. If there is no EDI section, the regular section's format will be used for the EDI format. See the last part of the example below.

The first line in a section may be a record type or a list of record types. If there are no record types there must be a section name in order for this section to be used. Subsequent lines contain the Label-COBOL format pairs in this section.

Record Types

There are several different record types, as listed below.

Record Types:

10 - Adjustment

20 - Cancel

30 - Current

40 - Rebill (one period)

50 - Rebill (multiple periods)

60 - Final bill

Record types 31, 41, 51, and 61 are detail records for the corresponding record type. For example, a record type of 31 indicates that it contains the details for a record type of 30, a current bill.

Sample CIS Transaction File

In the following example, the first three lines declare the setup of the control file. It declares that there is one common HEADER (with 6 Field Label-COBOL Format pairs) and two different format sections. There are three Field Label-COBOL Format pairs for format section one, two Field Label-COBOL Format pairs for section two, and so on. Format section one, in this example, shows how multiple record types can be associated with a format section. Format section two is only used for Bill Correct CANCEL transactions. Format sections three and four can handle special account requirements, with section four generating an EDI transaction record.

```
[HEADER];6
[SECTION];3
[SECTION];2
[SECTION];10
[SECTION];10
=====

[HEADER]
INTFC_CUST_ID;PIC X(21)
PWR_BILL_TY_CD; PIC X
TRANS_EFF_DATE; PIC X(10)
INTFC_RCD_TY; PIC XX
INTFC_ACC_ID; PIC X(21)
CUST_NAME; PIC X(21)
=====

[SECTION]
RecordType;10;30;40;50
BILL_HIST_PERIOD; PIC 99
READ_FROM_DT; PIC X(10)
READ_TO_DT; PIC X(10)
=====

[SECTION] CANCEL
BILL_HIST_PERIOD; PIC 99
READ_FROM_DATE; PIC X(10)
=====

[SECTION] SPECIAL_ACCT
BILL_HIST_PERIOD; ACCT1 - ACCT1 appears as is
READ_FROM_DATE; PIC X(10)
...
=====

[SECTION] SPECIAL_ACCT_ED
BILL_HIST_PERIOD; ACCT1EDI - ACCT1EDI appears as is
READ_FROM_DATE; PIC X(10)
...
=====
```

This control file will be read once and the information cached throughout the session. New changes to the control file will take effect upon restarting Oracle Utilities Billing Component.

Output for Record types 31, 41, 51, and 61 have special considerations:

- Data Values for each Field Label are taken directly from the Billing Engine's Report Window for each account (except account header). Hence NOTE 6 (above) does not apply for this case.
- Each line in the Report Window (except the report header) is a Data Value for the Field Label. There will be padding of blank lines, if there are less than 70 lines in the Report Window.
- Each Data Value will be blank padded to a maximum of 68 characters. Data Values with more than 68 columns will be truncated.
- Semicolon (;) is **not** allowed in Report Window, since semicolon is used as a delimiter.
- If the Billing Engine generates more than 70 lines of Data Values for an account, these Data Values will be split into two separate output records. The record header for these two records will be the same since they are for the same record. Blank lines appearing in Billing Engine's Report Window are considered valid Data Values.

Print Detail

There are four levels of detail bill printing available for the user to select: **Default**, **None**, **Print Normal Detail**, and **Print All Detail**.

Level	Description
Default:	See documentation for Automatic Billing for a detailed explanation.
None:	No CIS transactional record will be generated upon clicking Approve menu item.
Print Normal:	<i>Default.</i> CIS record will be generated without detailed billing information. Detail information of interval data results (such as MAXDATE, TOTAL) will not be displayed.
Print All:	CIS record generated along with the information from the Bill Calculation section of the bill report. This option will display the details of interval data.

The default is Print Normal Detail. In this scenario, detail information of interval data results such as MAXDATE, TOTAL will not be displayed. To display the details of interval data, Print All Detail should be checked. If the Print None option is selected, there will be no print transaction records written to the output file.

Chapter 10

Customizing the Energy Information Platform

This chapter describes how to customize the Oracle Utilities Energy Information Platform user interface, including:

- **Overview**
 - **What Sort of Customization Is Possible?**
 - **Basic File and Directory Structure**
 - **Customization Ground Rules**
- **Creating Custom Menus and Screens**
 - **Adding Custom Menu Items**
 - **Securing Custom Menu Items**
 - **Calling Existing Screens**
 - **Using the Data Navigator API**
 - **Creating Custom Pages**

Overview

Before you begin customizing the Energy Information Platform, it's important to understand some basic concepts about how the Oracle Utilities web-based software functions, and to note some ground rules that should be observed when adding customized files.

What Sort of Customization Is Possible?

Customizing the Energy Information Platform allows you to extend the user interface to meet your specific business needs. This includes the ability to create securable menu items that open custom-coded or Oracle Utilities-supplied pages

Basic File and Directory Structure

The functionality of the Energy Information Platform web framework is based on files in three directories:

- **C:\LODESTAR\Bin:** contains all of the Oracle Utilities-created executables, DLLs, and supporting files required to run the Oracle Utilities applications. This also includes permissions schemas (permissions.[xxx].XSD) which defines the privileges available through security.
- **C:\LODESTAR\CFG:** contains all the configuration files (lssecure.cfg.xml, lsreportmonitor.cfg.xml, lsschdlr.cfg.xml, etc.) used by Oracle Utilities applications.
- **C:\LODESTAR\Web** and **C:\LODESTAR\Web\classic:** contains all of the Oracle Utilities-created web files (ASP, HTML, XML, XSD, XSL, CCS, etc.) and images required to run Oracle Utilities web-based applications. A core set of folders exist that represent the functionality of the Energy Information Platform. In addition, folders corresponding to specific installed products (i.e. "be" for Oracle Utilities Billing Component, "fme" for Oracle Utilities Receivables Component) also exist in this directory.

Customization Ground Rules

To ensure compatibility with future upgrades, it's critical that you adhere to the following ground rules when customizing the Energy Information Platform user interface:

- No Oracle Utilities-created files or directories should be modified in any way or deleted.
- New menus, screens, and other supporting files should be created in custom directories (folders) only.
- Whenever possible, custom folders should be named such that they appear last alphabetically within the C:\LODESTAR\Web directory. One way to ensure this is to name your custom folder "zcustom" or something similar.
- All custom web functionality and corresponding folders should be located in the C:\LODESTAR\Web directory (described above) on the web server.
- When creating custom content based on existing functionality, create copies of existing Oracle Utilities-supplied files, and edit as necessary.
- When developing custom files (or editing existing files), make changes incrementally, and save results periodically.
- Oracle Utilities's development standard for custom pages is ASP (Active Server Pages) with Javascript

A Note About Dictionary Files

Dictionary files (dictionary.dic) define the labels that appear on the user interface. These files are used by all custom files, including custom menu items, custom ASP or HTML screens, and the Security Administration user interface. Dictionary files must be located in the same folder as any custom files you create. Without proper dictionary files in place, all menu section names, division names, and item names will be appear in brackets ({ }).

Several sections of this chapter describe dictionary files are used by different parts of the software (menu item labels, screen labels, etc.).

Note: When you create or update a "dictionary.dic" file, you must restart Microsoft Internet Information Services (IIS) on the web server in order for the changes to take effect. You can do this from the Services Windows Control Panel (IIS Admin Service). Consult your Windows documentation or online help for more information about IIS.

Creating Custom Menus and Screens

One way to customize the Energy Information Platform user interface is to add custom menu items that open screens (either custom or Oracle Utilities-supplied) suited to your specific business needs. This includes:

- **Adding Custom Menu Items**
- **Securing Custom Menu Items**
- **Calling Existing Screens**
- **Using the Data Navigator API**
- **Creating Custom Pages**

Adding Custom Menu Items

Most customization of the Energy Information Platform involves creating custom menu items. Custom menu items provide your users with a method of accessing your custom screens or functions.

To add custom menu items, use the following steps:

1. Create a custom folder (such as “zcustom”) in the **C:\LODESTAR\Web\classic** folder. This is the folder where all your custom files should be located.
2. Create a text file called “menu.xml” in the custom folder. This file defines your custom menu items. See **“Menu.XML” File Structure** on page 10-4 for details about creating this file.
3. Create a text file called “dictionary.dic” in the custom folder. This file defines the labels for the custom menu items and other custom content. See **“Dictionary.dic” File Structure** on page 10-6 for details about creating this file.

Creating the Menu.XML File

The “menu.xml” file defines the custom menu items to be added to the Energy Information Platform user interface.

“Menu.XML” File Structure

The basic structure of the “menu.xml” file is as follows:

```
<suite>
  <section name="zCustomMenu">
    <division name="MenuDivision">
      <item name="MenuItem1" action="../classic/zcustom/MenuItem1.asp">
        <param name="X_PRODUCTCODE" value="CUST" />
      </item>
    </division>
  </section>
</suite>
```

where:

suite: is the root element of the “menu.xml” file, and can contain one or more menu sections.

section: is an element that defines a menu section and can contain one or more menu items or divisions. Section elements have the following attributes:

- **name:** is the name of the menu section. Examples of this include “Entity Management”, “Tools and Utilities”, and “Work Queues”. Top level menu items appear in alphabetical order based on the section name. Section names **cannot** have spaces in them. To ensure that your custom menu appears at the bottom of the CCS Menu, name it something like “zCustomMenu.”

division: is an optional element that defines a menu division, and can contain one or more items. Division elements have the following attributes:

- **name:** is the name of an optional menu division. This allows the creation of sub menus beneath a menu section. Examples of this include “Configure Adapter Rules” and “Oracle Utilities Transaction Management/Adapter Components” under the “Tools and Utilities” menu. Menu divisions appear in alphabetical order within a menu section based on the division name. Division names **cannot** have spaces in them.

item: is an element that defines an individual menu item. Item elements have the following attributes:

- **name:** is the name of an individual menu item. Examples of this include “Accounts” and “Customers” under the “Entity Management” menu, and “Business Rules” and “Rule Description Language” under the “Configure Adapter Rules” sub menu. Menu items appear in alphabetical order within a menu division based on the item name. Item names **cannot** have spaces in them.
- **action:** is the path and file name of the file to be opened by the menu item, in the following format:

```
../<path>/<filename>
```

where:

- **<path>** is the path (relative to the C:\LODESTAR folder) to the file to be opened by the menu item.
- **<filename>** is the name of the file to be opened by the menu item.

For example, the “MenuItem1” menu item in the sample file above (“../zcustom/MenuItem1.asp”) opens the “MenuItem1.asp” file in the “zcustom” folder.

Menu Items can also point to a URL that opens an external website.

parameter: is an element that defines an optional parameter associated with the menu item. Parameter elements have the following attributes:

- **name:** is the name of an optional parameter associated with the menu item. For example, the “View Reports” menu options that appear under various menu sections use the “X_PRODUCTCODE” parameter to specify which reports to display. If supplying parameters for menu items pointing to custom pages, the custom pages should use the parameters in some way. Parameter names **cannot** have spaces in them.
- **value:** is the value of the <PARAMETER_NAME> associated with the menu item. For example, the “View Reports” menu option under the “Work Queues” menu used the “X_PRODUCTCODE” parameter with a value of “WQ” (the Product Code for Work Queues).

Example

To create a custom menu that includes three menu items, one in one division, and two in another division, you would create the following in the “menu.xml” file.

```
<suite>
  <section name="zCustomMenu">
    <item name="MenuItem1" action="../classic/zcustom/MenuItem1.asp" />
    <division name="MenuDivision">
      <item name="MenuItem2" action="../classic/zcustom/MenuItem2.asp" />
      <item name="MenuItem3" action="../classic/zcustom/MenuItem3.asp" />
    </division>
  </section>
</suite>
```

In this example, the top level menu item (at the same level as “Entity Management” or “Tools and Utilities”) is called “zCustomMenu.” Beneath that would be a menu item called “MenuItem1”, and

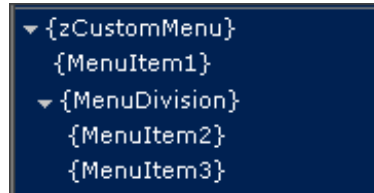
a menu division called “Menu2” under which would be two menu items (“MenuItem2” and “MenuItem3”).

Creating the Dictionary File

Dictionary files define the labels that will be visible on the user interface for the custom menu items defined in the “menu.xml” file. Dictionary files also define labels from other files, including custom ASP or HTML pages you might develop.

Dictionary files must be located in the same folder as the “menu.xml” file (or other custom files).

Without proper dictionary files in place, all menu section names, division names, and item names will be appear in brackets ({ }). For instance, if you didn’t create a dictionary.dic file for the custom menu described above, the menu items would look like the following on the user interface:



Note: When you create or update a “dictionary.dic” file, you must restart Microsoft Internet Information Services (IIS) on the web server in order for the changes to take effect. You can do this from the Services Windows Control Panel (IIS Admin Service). Consult your Windows documentation or online help for more information about IIS.

“Dictionary.dic” File Structure

The basic structure of the “dictionary.dic” file, when used with menu items, is as follows:

```
<dictionary>
  <MENU>
    <zCustomMenu ENU="Custom Menu"/>
    <MenuDivision ENU="Sub Menu"/>
    <MenuItem1 ENU="Menu Item One"/>
  </MENU>
</dictionary>
```

where:

dictionary: is the root element of the “dictionary.dic” file.

MENU: is an element which contains labels for menu sections, divisions, and items.

<MENU_NAME>: is an element that corresponds to the “name” attribute of a menu section, division, or item. In the above example, “CustomMenu” is the name of the menu section, “MenuDivision” is the name of the menu division, and “MenuItem1” is the name of a menu item.

<MENU_NAME> elements have the following attributes:

- **<LANGUAGE>:** a three letter ISO code that designates the language of the element. ENU is English, and is the default. When creating dictionary files or entries for other languages, the appropriate ISO code is used. For example, for French dictionary files, you would use “FRA”. For Italian dictionary files you would use “ITA”.

Example

To create labels for the custom menu in the above example, you would create the following in the “dictionary.dic” file.

```
<dictionary>
  <MENU>
    <zCustomMenu ENU="Custom Menu"/>
    <MenuDivision ENU="Sub Menu"/>
    <MenuItem1 ENU="Menu Item One"/>
    <MenuItem2 ENU="Menu Item Two"/>
  </MENU>
</dictionary>
```

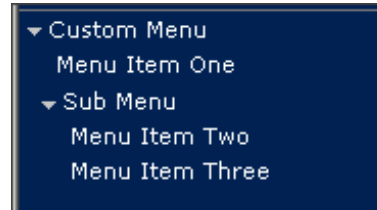
```

    <MenuItem3 ENU="Menu Item Three"/>
  </MENU>
</dictionary>

```

In this example, the top level menu item (at the same level as “Entity Management” or “Tools and Utilities”) would be labeled “Custom Menu.” Beneath that would be a menu item labeled “Menu Item One”, and a menu division labeled “Sub Menu” under which would be two menu items (labeled “Menu Item Two” and “Menu Item Three”, respectively).

On the Left Menu of the Energy Information Platform user interface, this custom menu would appear as follows:



Securing Custom Menu Items

Securing menu items allows you to restrict which users (or groups of users) can access your custom menu items (and the custom screens or functions the menu items point to).

To secure custom menu items, use the following steps:

1. Add security nodes to the “menu.xml” file. These nodes define the security privileges associated with your custom menu items. See **“Menu.XML” File Structure - Security Nodes** on page 10-8 for details about editing this file to add security nodes.
2. Create a permissions schema file in the **C:\LODESTAR\CFG** folder. This file defines the security nodes used to restrict access to your custom menu items. The nodes defined in this file appear in the Feature Privileges tree on the Privileges:Features screens (User and Group) on the Security Administration user interface. See **“Permission.*.XSD” File Structure** on page 10-9 for details about creating this file.
3. Create a permissions dictionary file in the **C:\LODESTAR\CFG** folder. This file defines the labels for the security nodes that will appear on the Security Administration user interface. See **“Permissions.*.dic” File Structure - Security** on page 10-12 for details about creating this file.

Adding Security Nodes to the “Menu.XML” File

Adding security nodes to the “menu.xml” file defines the security privileges associated with your custom menu items and enables securing those custom menu items on the Energy Information Platform user interface.

“Menu.XML” File Structure - Security Nodes

The structure of the “menu.xml” file changes slightly when securing menu items, as follows (changes noted in bold):

```
<suite>
  <section name="zCustomMenu" secure="//CUSTOMMENU">
    <item name="MenuItem1" action="../classic/zcustom/MenuItem1.asp" secure="//CUSTOMMENU/MENUIITEM1">
      <param name="PRODUCTCODE" value="CUST" />
    </item>
  </section>
</suite>
```

where:

secure: is an attribute on a menu section or item that specifies the security nodes associated with the menu section or item. The value specified for this attribute represents the xml structure in the Permissions schema (*.XSD) file used to define the security nodes associated with the menu items (see **Creating Permissions Schema Files** on page 10-9 for more information about permission schema files). In the above example, MENUITEM1 is a node of CUSTOMMENU (which is a node of PERMISSIONS). The format for this attribute is as follows:

```
//<section_node>[/<division_node>][/<item_node>]
```

where:

- **<section_node>** is the name of the security node associated with the menu section.
Example: **//CUSTOMMENU**
- **<division_node>** is the name of the security node associated with the menu division. Note that **<division_node>**s are only required if menu divisions are present.
Example: **//CUSTOMMENU/MENUDIVISION**
- **<item_node>** is the name of the security node associated with the menu item. Note that **<item_node>**s are only required for menu items.
Example: **//CUSTOMMENU/MENUIITEM1**

Note: Nodes must be in UPPER case.

Example

To add security nodes to the custom menu described above that includes three menu items, one in one division, and two in another division, you would edit the “menu.xml” file as follows (changed noted in **bold**).

```
<suite>
  <section name="zCustomMenu" secure="//CUSTOMMENU">
    <item name="MenuItem1" action="../classic/zcustom/MenuItem1.asp" secure="//CUSTOMMENU/MENUIITEM1">
      <division name="MenuDivision" secure="//CUSTOMMENU/MENUDIVISION">
        <item name="MenuItem2" action="../classic/zcustom/MenuItem2.asp" secure="//CUSTOMMENU/MENUDIVISION/MENUIITEM2">
        <item name="MenuItem3" action="../classic/zcustom/MenuItem3.asp" secure="//CUSTOMMENU/MENUDIVISION/MENUIITEM3">
      </division>
    </section>
  </suite>
```

In this example, the top level node is called “CUSTOMMENU.” Beneath that would be a node called “MENUITEM1”, and a division node called “MENUDIVISION” under which would be two nodes (“MENUITEM2” and “MENUITEM3”).

Creating Permissions Schema Files

Permissions schemas are hierarchical XML structures that define the security nodes used to restrict access to your custom menu items. The nodes defined in this file appear in the Feature Privileges tree on the Privileges:Features screens (User and Group) on the Security Administration user interface.

Permissions schema files use the following naming convention:

permissions.<CUSTOM_NAME>.xsd

where:

- <CUSTOM_NAME> is a unique name that identifies the schema file.

Note: When you create or update a “permissions.*.xsd” file, you must restart Microsoft Internet Information Services (IIS) on the web server in order for the changes to take effect. You can do this from the Services Windows Control Panel (IIS Admin Service). Consult your Windows documentation or online help for more information about IIS.

“Permission.*.XSD” File Structure

The structure of a permission schema file is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.2 U (http://www.xmlspy.com) by Al Bresnahan
(Lodestar Corp.) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="PERMISSIONS">
    <xs:complexType>
      <xs:all>
        <xs:element name="[SECTION_NODE]" minOccurs="0">
          <xs:complexType>
            <xs:all>
              <xs:element name="[DIVISION_NODE]" minOccurs="0">
                <xs:complexType>
                  <xs:all>
                    <xs:element name="[ITEM_NODE]" minOccurs="0" />
                  </xs:all>
                </xs:complexType>
              </xs:element>
            </xs:all>
          </xs:complexType>
        </xs:element>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

where:

PERMISSIONS: is the parent node. All security nodes are children of the PERMISSIONS node.

SECTION_NODE: is the node that corresponds to the menu section in the “menu.xml” file.

DIVISION_NODE: (optional) is a node that corresponds to a menu division in the “menu.xml” file.

ITEM_NODE: is a node that corresponds to a menu item in the “menu.xml” file.

How to create a permissions schema file:

To create a permissions schema file, use the following steps.

1. Create a text file in the C:\LODESTAR\CFG folder using the following naming scheme:

permissions.<CUSTOM_NAME>.xsd

where:

- **<CUSTOM_NAME>** is a unique name

2. Copy the contents of the **permissions.500_datasources.xsd** file from the C:\LODESTAR\Bin folder into this new file. The contents of this file are as follows:

```
<xs:schema elementFormDefault="qualified"
  attributeFormDefault="unqualified" xmlns:xs="http://www.w3.org/2001/
XMLSchema">
  <xs:element name="PERMISSIONS">
    <xs:complexType>
      <xs:all>
        <xs:element name="DATASOURCE" minOccurs="0"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

3. Edit the DATASOURCE node to match the section node specified in your “menu.xml” file. In the example above, this would be “CUSTOMMENU.”

```
<xs:schema elementFormDefault="qualified"
  attributeFormDefault="unqualified" xmlns:xs="http://www.w3.org/2001/
XMLSchema">
  <xs:element name="PERMISSIONS">
    <xs:complexType>
      <xs:all>
        <xs:element name="CUSTOMMENU" minOccurs="0"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

4. To add a division node (or nodes), include a [DIVISION_NODE] element under the node element in the file that specifies the menu section (“CUSTOMMENU”). This must be the exact division node as specified in the “menu.xml” file (“MENU DIVISION”). This element should use the following syntax:

```
...
  <xs:element name="CUSTOMMENU" minOccurs="0">
    <xs:complexType>
      <xs:all>
        <xs:element name="MENU DIVISION" minOccurs="0">
          </xs:all>
        </xs:complexType>
      </xs:element>
    </xs:complexType>
  </xs:element>
...
```

5. To add a menu node (or nodes), include an [ITEM_NODE] element under the [SECTION_NODE] or [DIVISION_NODE] element (as appropriate) for each menu item you wish to include. This must be the exact item node(s) as specified in the “menu.xml” file (“MENU ITEM2”). These elements should use the following syntax:

```
...
    <xs:element name="MENU DIVISION" minOccurs="0">
      <xs:complexType>
        <xs:all>
          <xs:element name="MENU ITEM2" minOccurs="0"/>
          <xs:element name="MENU ITEM3" minOccurs="0"/>
          ...
        </xs:all>
      </xs:complexType>
    </xs:element>
  </xs:complexType>
</xs:element>
```



```

        </xs:all>
      </xs:complexType>
    </xs:element>
  ...

```

Example

The permission schema file (permissions.custom.xsd) for the custom menu described above (that includes three menu items, one in one division, and two in another division), would be as follows (nodes denoted in **bold**):

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.2 U (http://www.xmlspy.com) by Al Bresnahan
(Lodestar Corp.) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="PERMISSIONS">
    <xs:complexType>
      <xs:all>
        <xs:element name="CUSTOMMENU" minOccurs="0">
          <xs:complexType>
            <xs:all>
              <xs:element name="MENUITEM1" minOccurs="0" />
              <xs:element name="MENUDIVISION" minOccurs="0">
                <xs:complexType>
                  <xs:all>
                    <xs:element name="MENUITEM2" minOccurs="0" />
                    <xs:element name="MENUITEM3" minOccurs="0" />
                  </xs:all>
                </xs:complexType>
              </xs:element>
            </xs:all>
          </xs:complexType>
        </xs:element>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

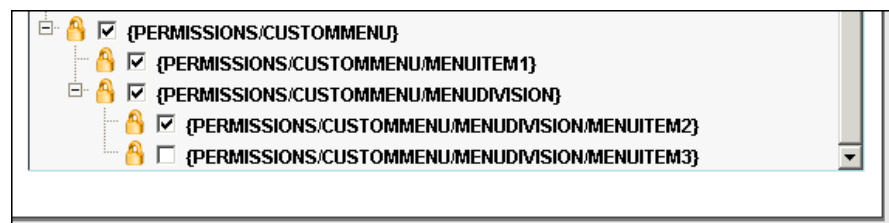
In this example, the top level node is called “CUSTOMMENU.” Beneath that would be a node called “MENUITEM1”, and a division node called “MENUDIVISION” under which would be two nodes (“MENUITEM2” and “MENUITEM3”).

Creating Permissions Dictionary Files

Just as menu items used dictionary files to define the labels that will be visible on the user interface, security nodes defined in permission schema files need dictionary files to define the labels for the security nodes that will appear on the Security Administration user interface.

Permission dictionary files must be located in the C:\LODESTAR\CFG folder.

Without proper dictionary files in place, all security nodes, including nodes for sections, divisions, and items will be appear in brackets ({ }). For instance, if you didn’t create a permissions dictionary.dic file for the permissions schema described above, the privileges would look like the following in the Feature Privileges tree on the Security Administration the user interface:



Permissions dictionary files use the following naming convention:

permissions.<CUSTOM_NAME>.dic

where:

- <CUSTOM_NAME> is the same (unique) name used when naming the permission schema file (“permissions.custom.dic”).

“Permissions.*.dic” File Structure - Security

The basic structure of the “permissions.*.dic” file used with permission schemas, is as follows:

```
<dictionary>
  <security>
    <PERMISSIONS>
      <CUSTOMMENU ENU="Custom Menu">
        <MENUITEM1 ENU="Menu Item One"/>
      </CUSTOMMENU>
    </PERMISSIONS>
  </security>
</dictionary>
```

where:

dictionary: is the root element of the “permissions.*.dic” file.

security: is an element that specifies that child elements are used by the Security Administration user interface.

PERMISSIONS: is an element that contains elements that define labels for menu sections, divisions, and items.

<NODE_NAME>: is an element that corresponds to a node defined in the permissions.*.xsd file. A <NODE_NAME> can be a menu section, division, or item. In the above example, “CUSTOMMENU” is the node name of the menu section, “MENUITEM1” is the node name of a menu item. <NODE_NAME> elements have the following attributes:

- **<LANGUAGE>:** a three letter ISO code that designates the language of the element. ENU is English, and is the default. When creating dictionary files or entries for other languages, the appropriate ISO code is used. For example, for French dictionary files, you would use “FRA”. For Italian dictionary files you would use “ITA”.

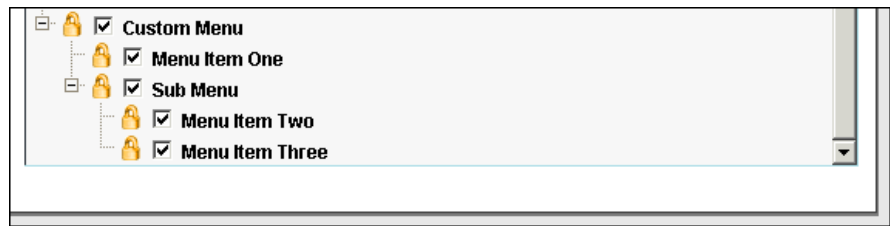
Example

To create labels for the security nodes in the above example , you would create the following file (“permissions.custom.dic”):

```
<?xml version="1.0"?>
<dictionary>
  <security>
    <PERMISSIONS>
      <CUSTOMMENU ENU="Custom Menu">
        <MENUITEM1 ENU="Menu Item One"/>
        <MENUDIVISION ENU="Sub Menu">
          <MENUITEM2 ENU="Menu Item Two"/>
          <MENUITEM3 ENU="Menu Item Three"/>
        </MENUDIVISION>
      </CUSTOMMENU>
    </PERMISSIONS>
  </security>
</dictionary>
```

In this example, the top level node (at the same level as “Entity Management” or “Tools and Utilities”) would be labeled “Custom Menu.” Beneath that would be a node labeled “Menu Item One”, and a division node labeled “Sub Menu” under which would be two item nodes (labeled “Menu Item Two” and “Menu Item Three”, respectively).

In the Feature Privileges tree on the Security Administration user interface, the security nodes would appear as follows:



Calling Existing Screens

Custom menu items don't necessarily have to open custom-coded screens. You can also create custom menus to open (or call) existing screens. Two common applications of this are reporting screens (including the Run Reports, View Reports, and Enter Report Parameters screens), and Data Navigator screens.

Basic Syntax

The basic syntax used to call pages is as follows:

```
...
<item name="<MENU_ITEM>" action="../<FOLDER>/<FILE>">
  <param name="<PARAMETER_NAME>" value="<PARAMETER_VALUE>" />
...
```

where:

- **<MENU_ITEM>** is the menu item used to call the page.
- **<FOLDER>** is the folder in the C:\LODESTAR\Web folder that contains the page to be called.
- **<FILE>** is the name of the page to be called.
- **<PARAMETER_NAME>** is the name of a parameter to be passed to the called page. You will almost always want to supply a parameter (or parameters) when calling pages, as it is the parameters which differentiate your custom menu item from the standard menu item that calls the same page.
- **<PARAMETER_VALUE>** is the value of the parameter to be passed to the called page. Every parameter **MUST** have an associated value.

Example

The following example defines custom menu items that open the Run Reports and View Reports screens for reports associated to the "CUST" product code:

```
<suite>
  <section name="zCustomMenu">
    <item name="CustRunReport" action="../classic/lsrcf/SearchResult.asp" >
      <param name="X_PAGENUMBER" value="1" />
      <param name="X_ORDER_BY" value="RPTNAME" />
      <param name="X_ORDER_DIRECTION" value="ASC" />
      <param name="X_PRODUCTCODE" value="CUST" />
      <param name="Mode" value="RUN" />
    </item>
    <item name="CustViewReport" action="../classic/lsrcf/SearchReportsResult.asp" >
      <param name="X_ROWPERPAGE" value="25" />
      <param name="X_PAGENUMBER" value="1" />
      <param name="X_ORDER_BY" value="RPTNAME" />
      <param name="X_ORDER_DIRECTION" value="ASC" />
      <param name="X_PRODUCTCODE" value="CUST" />
    </item>
  </section>
</suite>
```

The menu items in this example also use the following parameters:

- **X_ROWPERPAGE**: specifies how many records to display per page.
- **X_PAGENUMBER**: specifies which page of results to display (if more than one)
- **X_ORDER_BY**: specifies the column used to sort the records.
- **X_ORDER_DIRECTION**: specifies the direction in which the records are sorted (ASC - ascending, or DES- descending).
- **X_PRODUCTCODE**: specifies the product code (from the LODESTAR Product table) associated with the report templates and report instances to display.
- **Mode**: specifies the mode in which to display the “SearchResult.asp” screen (“RUN”).

Report Screens

Creating custom menu items that point to the reporting screens (Run Reports, View Reports, and Enter Report Parameters) allow you to run and view custom reports related to custom product codes directly from your custom menu.

Run Reports: The Run Reports screen is used to run reports defined using the Report Administration user interface.

The path and filename for the Run Reports screen is:

```
../lsrf/SearchResult.asp
```

Example:

```
...  
    <item name="CustRunReport" action="../classic/lsrf/SearchResult.asp"  
secure="//WQ/RPTRUN">  
        <param name="X_PRODUCTCODE" value="CUST" />  
        <param name="Mode" value="RUN" />  
    </item>  
...
```

The above menu item would open the Run Reports screen and display all report templates with a product code of “CUST”.

View Reports: The View Reports screen is used to view reports (records in the Report Instance table) run using the Report Framework.

The path and filename for the View Reports screen is:

```
../lsrf/SearchReportsResult.asp
```

Example:

```
...  
    <item name="CustViewReport" action="../classic/classic/lsrf/  
SearchReportsResult.asp" secure="//WQ/RPTVIEW">  
        <param name="X_ROWPERPAGE" value="25" />  
        <param name="X_PAGENUMBER" value="1" />  
        <param name="X_PRODUCTCODE" value="CUST" />  
    </item>  
...
```

The above menu item would open the View Reports screen and display all report instances with a product code of “CUST”.

Enter Report Parameters Screen: The Enter Report Parameters screen is used to enter report parameters for Crystal Reports and Oracle Utilities Rules Language (LSRate) Reports. This screen is dynamically created by the Report Framework when the user selects a report template from the Run Reports screen. Calling this screen directly allows you to create a menu item that opens the Enter Report Parameters for a specific report immediately, without causing the user to first visit the Run Reports screen.

The path and filename for the Enter Report Parameters screen is:

```
../lsrf/parameters.asp
```

This screen requires the “RptName” parameter be passed to specify the report to be run. The Value of this parameter must be the name of the Report Template on which the report to be run is based.

Example:

```
...
    <item name="Run_Custom_Reports" action="../classic/lsrf/parameters.asp">
        <param name="RptName" value="WQAging" />
        <param name="X_PRODUCTCODE" value="WQ" />
    </item>
...
```

The above menu item would open the Enter Report Parameters screen for the Work Queue Aging report.

Other Screens

You can call any screen from a custom menu. The table below lists some of the common screens in the Energy Information Platform and the folder and filename for the corresponding files.

Screen Name	Folder and Filename
Accounts	../classic/viewaccount/Search.asp
Customers	../classic/viewcustomer/Search.asp
Recorders	../classic/idg/Search.asp
Meters	../classic/viewmeter/Search.asp
Interval Data Manager Cut Search	../classic/ide/CutSearch.asp
Interval Data Manager Workset	../classic/ide/Workset.asp
Search Work Queue Items	../classic/wq/Search.asp
Search Work Queue Closed Items	../classic/wq/ClosedItemSearch.asp

Using the Data Navigator API

Data Navigator allows users to search, view, and edit database records and tables in the Oracle Utilities Data Repository. The Data Navigator API provides a mechanism to create custom menu items that open Data Navigator screens without requiring the user to access the Data menu.

Some examples of how you might use the Data Navigator API include:

- Create a menu item that opens a search screen to search records in a custom (or existing) table.
- Create a menu item that opens a search screen to search records based on certain criteria (associated to a particular Operating Company, Jurisdiction, etc.).
- Create a menu item that lists all records in a table that match a certain criteria (associated to a particular Operating Company, Jurisdiction, etc.).
- Create a menu item that allows a user to edit a specific record.
- Create a menu item that allows a user to add a record to a table with pre-populated values.

Basic Syntax

The path and filename for all Data Navigator screens is:

```
../classic/meta/QueryInit.asp
```

The basic syntax for calling Data Navigator screens is as follows:

```
...
<item name="SearchAccounts" action="../classic/meta/QueryInit.asp" >
  <param name="Command" value="Query" />
  <param name="TableName" value="Account" />
</item>
...
```

where:

- **Command:** is a parameter that specifies the type of screen to open.
- **TableName:** is a parameter that specifies the name of the table in the Oracle Utilities Data Repository to be accessed.

Note: Data Navigator API menu items are similar to other custom menu items in that they require entries in the “dictionary.dic” file (see **Adding Custom Menu Items** on page 10-4) and can be secured (see **Securing Custom Menu Items** on page 10-7).

Parameters

Unlike other screens which do not always require parameters, Data Navigator screens always require parameters to (minimally) define the type of screen to display and the table to access. Data Navigator screens can use the following parameters:

- **TableName:** [Required] specifies the table to be accessed.
- **Command:** specifies the type of screen to open. This can be one of the following:
 - **Query:** opens a Data Navigator search screen for the specified table (from the TableName parameter) (default)
 - **List:** opens a Data Navigator list screen for the specified table
 - **Edit:** opens a Data Navigator edit screen for the specified table
 - **Add:** opens a Data Navigator add screen for the specified table
- **ROWPERPAGE:** sets the value of the “Page Length” field on search screens. Specifies how many records to display per page on list screens.
- **PAGENUMBER:** specifies which page of results to display (if more than one) on list screens.
- **SORT_BY:** sets the value of the “Sort By” drop-down list on search screens. Specifies the column used to sort the records on list screens.
- **SORT_ORDER:** specifies the direction in which the records are sorted on list screens. Can be either “ASC” (ascending, default) or “DES” (descending).
- **CASESENSITIVE:** specifies if search criteria is case-sensitive. Can be either “YES” (default) or “NO”.
- **BoolOp:** specifies how the search criteria are to be combined. Can be either “AND” (default) or “OR”.
- **O_<COLUMN_NAME>:** operand to be passed to the column specified by <COLUMN_NAME>. This can be one of the following:

Operand	Explanation
EQ	The column value is Equal to the “X_” parameter.

Operand	Explanation
NE	The column value is Not Equal to the “X_” parameter.
LT	The column value is Less Than the “X_” parameter.
LE	The column value is Less Than or Equal to the “X_” parameter.
GT	The column value is Greater Than the “X_” parameter.
GE	The column value is Greater than or Equal to the “X_” parameter.
LIKE	The column value matches the “X_” parameter.
NOTLIKE	The column value does not match the “X_” parameter.
SOUNDX	The column value is similar to the “X_” parameter.
NOTSOUNDX	The column value is not similar to the “X_” parameter.
NULL	The column is Null .
NOTNULL	The column is Not Null .

For example, to set the operand for the Jurisdiction (JURISCODE) column to “equals” (EQ), you would create the following parameter:

```
<param name="O_JURISCODE" value="EQ" />
```

- **X_<COLUMN_NAME>**: value of the column specified by <COLUMN_NAME>. For example to set the value of the Jurisdiction (JURISCODE) column to “LODESTAR”, you would create the following parameter:

```
<param name="X_JURISCODE" value="LODESTAR" />
```

- **D_NoHistory**: specifies if the “History” section appears. Can be either “true” or “false” (default).
- **D_NoChildren**: specifies if the “Children” section appears on the Edit and Add screens. Can be either “true” or “false” (default).
- **D_NoNav**: specifies if the ellipsis ([...]) column appears on list screens. Can be either “true” or “false” (default).
- **D_NoSel**: specifies if the checkox selection column appears on list screens. Can be either “true” or “false” (default).
- **D_Exclude**: specifies individual column(s) to hide. To specify multiple columns, list each column to be hidden in the value attribute, separated by a space. For example, to hide the User ID (LSUSER) and Timestamp (LSTIME) columns, you would create the following parameter:

```
<param name="D_Exclude" value="LSUSER LSTIME" />
```

A Note About Parameters: Parameters supplied for a custom menu item using the Data Navigator API remain in effect until the user selects another menu item. For example, a user selects a menu item that opens a search screen which includes the “D_NoHistory” and “D_NoChildren” parameters (both set to “true”). After the user initiates the search, the History and Children sections will both be hidden on each screen as the user navigates from screen to screen (i.e. from the list screen to a specific record). However, if/when the user selects a different menu item (one that does NOT include the same parameters), the History and Children sections would appear as normal.

Examples

The following examples illustrate the types of menu items possible using the Data Navigator API. These examples can also be found in the C:\LODESTAR\Web\meta\xMenu.xml file on the web server:

Search Example 1: Open a search screen for the Account table

```
...
  <item name="Query" action="../../../classic/meta/QueryInit.asp" >
    <param name="Command" value="Query" />
    <param name="TableName" value="ACCOUNT" />
  </item>
...
```

Search Example 2: Open a search screen for the Account table. When displaying search results, display 13 records per page, starting on the second page of results. Sort the search results by the STARTTIME column in descending order. Search criteria should be case-insensitive, and should be combined with “OR.”

```
...
  <item name="QueryOptions" action="../../../classic/meta/QueryInit.asp">
    <param name="Command" value="Query" />
    <param name="TableName" value="ACCOUNT" />
    <param name="ROWPERPAGE" value="13" />
    <param name="PAGENUMBER" value="2" />
    <param name="SORT_BY" value="STARTTIME" />
    <param name="SORT_ORDER" value="DESC" />
    <param name="CASESENSITIVE" value="NO" />
    <param name="BoolOp" value="OR" />
  </item>
...
```

Search Example 3: Open a search screen for the Account table, with the value of the Jurisdiction column set equal to “TX.”

```
<item name="QueryParams" action="../../../classic/meta/QueryInit.asp" >
  <param name="Command" value="Query" />
  <param name="TableName" value="ACCOUNT" />
  <param name="O_JURISCODE" value="EQ" />
  <param name="X_JURISCODE" value="TX" />
</item>
...
```

List Example 1: Open a list screen for the Account table.

```
...
  <item name="List" action="../../../classic/meta/QueryInit.asp"
    <param name="Command" value="List" />
    <param name="TableName" value="ACCOUNT" />
  </item>
...
```

List Example 2: Open a list screen for the Account table that displays 13 records per page, and starts on the second page. Sort the list by the STARTTIME column in descending order.

```
...
  <item name="ListOptions" action="../../../classic/meta/QueryInit.asp">
    <param name="Command" value="List" />
    <param name="TableName" value="ACCOUNT" />
    <param name="ROWPERPAGE" value="13" />
    <param name="PAGENUMBER" value="2" />
    <param name="SORT_BY" value="STARTTIME" />
    <param name="SORT_ORDER" value="DESC" />
  </item>
...
```


Edit Example 1: Open an edit screen for Account “800001.”

```
...
    <item name="Edit" action="../../classic/meta/QueryInit.asp" >
        <param name="Command" value="Edit" />
        <param name="TableName" value="ACCOUNT" />
        <param name="O_ACCOUNTID" value="EQ" />
        <param name="X_ACCOUNTID" value="800001" />
    </item>
...
```

Edit Example 2: Open an edit screen for Account “800001.” When navigating from this page, list screens should display 13 records per page, and start on the second page. Sort lists by the STARTTIME column in descending order. On search screens, search criteria should be case-insensitive, and should be combined with “OR.”

```
...
    <item name="EditOptions" action="../../classic/meta/QueryInit.asp" >
        <param name="Command" value="Edit" />
        <param name="TableName" value="ACCOUNT" />
        <param name="ROWPERPAGE" value="13" />
        <param name="PAGENUMBER" value="2" />
        <param name="SORT_BY" value="STARTTIME" />
        <param name="SORT_ORDER" value="DESC" />
        <param name="CASESENSITIVE" value="NO" />
        <param name="BoolOp" value="OR" />
        <param name="O_ACCOUNTID" value="EQ" />
        <param name="X_ACCOUNTID" value="800001" />
    </item>
...
```

Add Example: Open as add screen for the Account table.

```
...
    <item name="AddAccount" action="../../classic/meta/QueryInit.asp" >
        <param name="Command" value="Add" />
        <param name="TableName" value="ACCOUNT" />
    </item>
...
```

Creating Custom Pages

You can also create entirely new pages and incorporate them into the Energy Information Platform user interface.

Custom pages can be created using any non-proprietary language viewable through Microsoft Internet Explorer, such as ASP (Active Server Pages), HTML (Hypertext Markup Language), or XML (Extensible Markup Language).

Specifics regarding coding in the above languages is beyond the scope of this documentation. However, the following are some important things to consider when creating custom pages:

- Oracle Utilities's development standard for custom pages is ASP (Active Server Pages) with Javascript.
- When possible, use an installed page as a template for your custom pages. For example, if you need to create a search page, use the "Search.asp" page from either the "viewaccount", "viewcustomer", or "viewmeter" folders in the C:\LODESTAR\Web folder on the web server.
- Like all other custom files, the files used by custom pages (such as ASP files, Java Script files, HTML files, or XML files) should be created in a custom folder (such as "zcustom") in the C:\LODESTAR\Web folder on the web server.
- Custom pages may require entries in a dictionary file ("dictionary.dic"). These entries should be included within a custom section of the XML structure of the dictionary file. For example, if creating a "Search Retailers" screen, your dictionary file should contain a section (such as "SEARCHRETAILERS") that contains entries for this screen (see example below).

```
<dictionary>
  <MENU>
    ...
  </MENU>
  <SEARCHRETAILERS>
    ...
  </SEARCHRETAILERS>
</dictionary>
```

Oracle Utilities strongly recommends you work with Oracle Utilities staff when creating custom pages.

Part Two

System Administration

Part Two describes system administration of Oracle Utilities Energy Information Platform, and contains the following chapters:

- **Chapter 11: Oracle Utilities Energy Information Platform Services**
- **Chapter 12: Energy Information Platform Security**
- **Chapter 13: Energy Information Platform Reporting Framework**
- **Chapter 14: Troubleshooting**

Chapter 11

Oracle Utilities Energy Information Platform Services

This chapter describes a number of features, services, and background processes employed by the Oracle Utilities Energy Information Platform, including:

- **Interval Data Storage and Access**
- **Energy Information Platform Report Monitor**
- **Energy Information Platform Report Scheduler**
- **Energy Information Platform Logging Framework**
- **Database Auditing**

Interval Data Storage and Access

One of the key features of the Energy Information Platform is interval data storage and access. This section describes how interval data is stored and accessed using the Oracle Utilities Energy Information Platform, including:

- **Standard Interval Data Storage**
- **Enhanced Interval Data Storage**
- **Interval Data Access**
- **Interval Data Versioning**

Standard Interval Data Storage

Standard interval data storage is the type of interval data storage used in legacy versions of Oracle Utilities (formerly LODESTAR) applications. In standard interval data storage, interval data is always related to a Recorder-Channel combination. This section describes standard interval data storage in the Oracle Utilities Data Repository, including:

- **Recorder/Channel Tables**
- **Interval Data Tables**
- **Staging Tables**
- **Reporting Tables**
- **Interval Data Values**
- **Interval Data Status Codes**
- **Units of Measure**

Recorder/Channel Tables

Recorder/Channel tables define the parent records of interval data cuts stored in the Oracle Utilities Data Repository.

Recorder Table

Records in the Recorder table represent individual interval data recorders, devices used to record interval data. Records in the Recorder table are the required parent of records in the Channel table.

Channel Table

Records in the Channel table represent a specific type of data (often a unit of measure) recorded by a recorder. Records in the Channel table are the required parent of interval data cuts.

Interval Data Tables

Standard interval data storage involves storing interval data in a specific set of tables in the Oracle Utilities Data Repository. Each set of interval data tables includes:

- A header table (stores header information for each cut)
- A data table (stores interval data values and status codes for each cut)
- A validation messages table (stores validation messages for each cut)
- An edit trails table (stores edit trails for each cut)

The default set of interval data tables supplied with the base database schema include:

- **Channel Cut Header Table (LSCHANNELCUTHEADER)**
- **Channel Cut Data (LSCHANNELCUTDATA)**

- **Channel Cut Validation Message (LSCHANNELCUTVMSGs)**
- **Channel Cut Edit Trail (LSCEDITRAILS)**

Additional table sets can be added to the Oracle Utilities Data Repository through meta-data modifications.

Channel Cut Header Table (LSCHANNELCUTHEADER)

The Channel Cut Header table stores header information for individual interval data cuts. Records in this table contain the following information:

- **Recorder ID:** The parent Recorder ID of the cut.
- **Channel Number:** The parent Channel Number of the cut.
- **Start Time:** The start time of the cut.
- **Stop Time:** The stop time of the cut.
- **SPI:** The number of seconds-per-interval of each interval in the cut.
- **UOM Code:** The unit of measure recorded in the cut. See **Units of Measure** on page 11-7 for more information about units of measure.
- **DST Participant:** A flag that designates whether or not the parent recorder observes Daylight Saving Time. Valid values are “Y” (Yes) and “N” (No).
- **Time Zone:** The time of the cut, measured in half-hour increments west of GMT (Greenwich Mean Time).
- **Interval Count:** The number of intervals in the cut.
- **Channel Cut Time Stamp:** A timestamp that indicates when the cut was first imported into the Oracle Utilities Data Repository.
- **Origin:** The origin of the cut. Must be one of: M (metered), P (profiled), C (computed), or S (Oracle Utilities Load Analysis Statistic).
- **Start Reading:** The meter start reading for the cut.
- **Stop Reading:** The meter stop reading for the cut.
- **Meter Multiplier:** The meter multiplier for the cut. Used to calculate consumption for the cut based on the Start Reading and Stop Reading, based on the following formula:
Consumption = (((Stop Reading - Start Reading) * Meter Multiplier) + Meter Offset)
- **Meter Offset:** The meter offset for the cut. Used to calculate consumption for the cut based on the Start Reading and Stop Reading, based on the following formula:
Consumption = (((Stop Reading - Start Reading) * Meter Multiplier) + Meter Offset)
- **Pulse Multiplier:** The pulse multiplier for the cut. Used to convert pulse readings into engineering units, based on the following formula:
Engineering Unit = (Pulse Value * Pulse Multiplier) + Pulse Offset
- **Pulse Offset:** The pulse multiplier for the cut. Used to convert pulse readings into engineering units, based on the following formula:
Engineering Unit = (Pulse Value * Pulse Multiplier) + Pulse Offset
- **Edited:** A flag that indicates if the interval data cut has been edited.
- **Internal Validation:** A flag that indicates if the cut is internally valid.
- **External Validation:** A flag that indicates if the cut is valid in relation to the cuts immediately preceeding and following the cut
- **Merge Flag:** A flag that indicates if the interval data cut has been merged with another cut.
- **Delete Flag:** A flag that indicates if the interval data cut has been deleted.

- **Val Flag E:** A flag used by the Oracle Utilities Load Analysis validation programs to indicate internal test validation failure.
- **Val Flag I:** A flag used by the Oracle Utilities Load Analysis validation programs to indicate internal test validation failure.
- **Val Flag O:** A flag used by the Oracle Utilities Load Analysis validation programs to indicate internal test validation failure.
- **Val Flag N:** A flag used by the Oracle Utilities Load Analysis validation programs to indicate internal test validation failure.
- **TK Written Flag:** A flag that indicates if the cut was written using the Oracle Utilities Load Analysis ToolKit.
- **Direction:** The power flow direction of the cut (Received or Distributed), relative to the grid. Distributed indicates power flowing from the grid to the meter. Received indicates power flowing from the meter to the grid. When this is set to R (Received), values and totals are set to negative values when accessed via Oracle Utilities Rules Language.
- **Accept/Reject Status:** The accept/reject status of the cut.
- **Translation Time:** The translation time of the cut
- **Descriptor:** A description of the cut.
- **Edited by Rules Schedule:** A flag that designates if the cut has been edited by Oracle Utilities Rules Language.
- **Time Zone Standard Name:** The Time Zone Standard Name for the time zone in which the cut was recorded. The value must be one of EST, CST, MST, PST, or be defined in the LSCALENDAR.XML configuration file (if present).
- **Population:** The statistical population associated with the cut. Used by Oracle Utilities Load Analysis.
- **Weight:** The statistical weight associated with the cut. Used by Oracle Utilities Load Analysis.

Channel Cut Data (LSCHANNELCUTDATA)

The Channel Cut Data table stores interval values and status codes for individual interval data cuts. Records in this table contain the following information:

- **Channel Cut Recorder ID:** The parent Recorder ID of the cut.
- **Channel Cut Channel Number:** The parent Channel Number of the cut.
- **Channel Cut Start Time:** The start time of the cut.
- **Value Codes:** The interval values and status codes of the cut, stored in a binary format. See **Interval Data Values** on page 11-6 for more information about this format.

Channel Cut Validation Message (LSCHANNELCUTVMSG)

The Channel Cut Validation Message table stores validation messages for individual interval data cuts. Validation messages are created either by Oracle Utilities Rules Language, or the Oracle Utilities Load Analysis application. Records in this table contain the following information:

- **Channel Cut Recorder ID:** The parent Recorder ID of the cut.
- **Channel Cut Channel Number:** The parent Channel Number of the cut.
- **Channel Cut Start Time:** The start time of the cut.
- **Validation Messages Sequence:** The number of the validation message. Validation messages are stored in sequential order.
- **Message Text:** The text of the validation message.

Channel Cut Edit Trail (LSCEDITRAILS)

The Channel Cut Edit Trail table stores edit trails for individual interval data cuts. Edit trails are created either by Interval Data Manager or the Oracle Utilities Load Analysis application. Records in this table contain the following information:

- **Channel Cut Recorder ID:** The parent Recorder ID of the cut.
- **Channel Cut Channel Number:** The parent Channel Number of the cut.
- **Channel Cut Start Time:** The start time of the cut.
- **Edit Number:** The number of the edit message.
- **Edit Text:** The text of the edit message.

Version Tables

In addition to the above interval data tables, the Oracle Utilities Data Repository also includes version tables used to store historical versions of interval data cuts. These tables are used when interval data versioning is enabled (see **Interval Data Versioning** on page 11-21 for more information) and include the following:

- **Channel Cut Header Version (LSCHVERSION):** Stored versions of Channel Cut Header records.
- **Channel Cut Data Version (LSCDVERSION):** Stores versions of Channel Cut Data records.
- **Channel Cut Validation Message Version (LSCVMSGVERSION):** Stored versions of Channel Cut Validation Message records.

Staging Tables

Staging tables are optional tables used as a means of temporary storage interval data. Staging tables might be used to when data is initially imported into the database piece-meal, or in cases where the data is imported over an extended period of time. Once data is in a staging table, it can be moved into the appropriate enhanced interval data table using Rules Language.

Staging tables are unique in that they can store only a fixed number of intervals. For each interval to be stored, the staging table has a pairing of Value and Status columns, which store the interval value and status code for each interval. For example, if a staging table was going to be used to store up to 1 day of 15-minute data, it might have 100 Value/Status column pairs (1 for each of 96 intervals, plus 4 to accommodate the extra DST hour in the Fall).

Note: It is an error if you attempt to save an interval data handle to a staging table that does not have sufficient number of Value/Status column pairs.

There is one staging table used with standard interval data storage, the Interval Data Staging table.

Reporting Tables

Reporting tables are optional tables used as a means of temporarily storing interval data for reporting and/or data export.

For standard interval data storage, there are two reporting tables, the LSRDINTDHEADER table, and the LSRFINTDVALUES table. These tables store interval data as individual cuts, similar to the way interval data is stored in the Channel Cut Header and Channel Cut Data tables, but in a format that is accessible by Crystal Reports. For each cut stored in these tables, there is one record in the LSRFINTDHEADER table, and one record for each interval in the cut in the LSRFINTDVALUES table.

LSRFINTDHEADER Table

Records in the LSRFINTDHEADER table contain header information for each interval data cut, and include the following information:

- **UIDINTDHEADER:** The unique identifier for the cut in the table. This is also the primary key of the record.
- **RECORDER:** The Recorder ID of the cut.
- **CHANNEL:** The channel number of the cut.
- **SPI:** The Seconds-per-interval (SPI) of the cut.
- **UOMCODE:** The Unit-of-Measure (UOM) code of the cut.
- **DSTPARTICIPANT:** The cut's DST Participant flag.
- **ORIGIN:** The cut's Origin flag.
- **TZSTDNAME:** The Time Zone Standard Name for the cut.

LSRFINTDVALUES Table

Records in the LSRFINTDVALUES table store individual interval values for each interval data cut. Each record in this table is the child of a corresponding record in the LSRFINTDHEADER table, and includes the following information:

- **UIDINTDHEADER:** The unique identifier of the cut, from the LSRFINTDHEADER table.
- **DATETIME:** The interval timestamp.
- **VALUE:** The interval value.
- **STATUS:** The interval status code.

Interval Data Values

The Value Codes (VALUECODES) column of the Channel Cut Data table contains interval data values, interval data status codes in a binary format. The number (or count) of intervals and status code pairs in this column is stored in the Interval Count column of the Channel Cut Data table. The space designated for this binary data is determined by the following calculation:

$$(T + 1) * (9 \text{ bytes}) = \text{length of binary column}$$

where:

- T = Total intervals (from INTERVALCOUNT column)

Each interval requires 9 bytes because each consists of an 8 byte double for the interval data value, and a 1 byte ASCII character for the interval data status code.

Interval Values

Interval data values are stored contiguously. The first interval data value is contained in bytes 1 - 8, the second interval data value is contained in bytes 9 - 16, and the nth interval data value is contained in bytes $((8 * n) - 7)$ through bytes $(8 * n)$. After the last interval data value are 8 bytes of unused space.

Interval Status Codes

Interval data status codes are stored contiguously following these 8 bytes. The first interval data status is stored in byte $((8 * (T + 1)) + 1)$, the second interval data status is in byte $((8 * (T + 1)) + 2)$, and the nth interval data status is stored in byte $((8 * (T + 1)) + n)$.

After the last interval data status code there is one byte of unused space.

Interval Data Status Codes

Interval data status codes indicate the status of each of the intervals within the cut. The Oracle Utilities applications use the following set of status codes:

Status Code	Classification of interval data
<i>(blank)</i>	Normal
A	Normal, alternate-recorded (e.g., hand-entered)
J	Data inserted by Oracle Utilities Load Analysis to correct outage
L	Default for data modified by Oracle Utilities Load Analysis Load Data Editor
M	Splice peak interval (estimated value)
N	Interruptible or curtailable load
P	Inserted outage
Q	Corrected outage
X	Cuts resulting from merging invalid data or from unrecognized status codes
Y	Reserved
1	Uncorrected outage
2	Non-normal (usually timing-pulse defects)
5	Aggregated interval used in rolling format with partially missing or unavailable data
7	Marged, aggregated, or transformed interval with partially missing data
9	Missing

Units of Measure

Interval data cuts measure data recorded in specific units of measure, defined in the Unit of Measure table. The Oracle Utilities Data Repository includes the following pre-defined unit of measure records:

CODE	DESCRIPTION	AGGREGATION TECHNIQUE
01	kWh	Sum
02	kW	Average
03	kVARh	Sum
04	kVAH	Sum
05	Temperature (°F)	Average
06	KQD	Average
07	V ² H (PTP)	Sum
08	kQh	Sum
09	kQh (45 degrees)	Sum

CODE	DESCRIPTION	AGGREGATION TECHNIQUE
10	I ² H	Sum
11	Volts	Average
12	Amps	Average
13	Temperature (°C)	Average
14	Dewpoint	Average
15	Amplitude	Average
16	Miscellaneous	Sum
17	Minute Run Time (MRT)	Sum
18	Wind Velocity (Cm/Sec)	Average
19	Fraction V ² H (PTN)	Average
20	Percent	Average
21	Flow	Average
22	kVAR	Average
23	kVA	Average
24	kVA Ratio	Average
25	Power Factor	Average
26	Hertz	Average
27	Feet	Average
28	Minutes	Sum
29	On / Off (Tap position)	Average
30	Inches	Average
31	Individual kWh	Sum
32	kWh r	Sum
33	Individual Totalized kVARh	Sum
34	kVARh r	Sum
35	Individual Totalized Temperature (°F)	Average
36	kVAh r	Sum
37	IND V ² H (Individual totalized V ² H)	Sum
38	IND kQh (Individual totalized kQh)	Sum
39	KQH r	Sum
40	Miscellaneous	Average
41	IND Volts (Individual totalized Volts)	Average

CODE	DESCRIPTION	AGGREGATION TECHNIQUE
42	IND Amps (Individual totalized Amperes)	Average
43	IND Temperature (°C) (Individual totalized temperature, degrees Celsius)	Average
44	Mw	Sum
45	MVAR	Average
46	MVA	Average
47	IND MRT (Individual totalized MRT)	Sum
48	IND CMS (Individual totalized CMS)	Average
49	Run Hours	Sum
50	Equivalent Full Load Hours	Sum
51	KWH-OUT	Sum
52	KW-OUT	Average
53	KVARH-OUT	Sum
54	KVAH-OUT	Sum
55	KQH-OUT	Sum
56	LEAD-KVARH	Sum
57	LEAD-KVARH-OUT	Sum
58	LAG-KVARH	Sum
59	LAG-KVARH-OUT	Sum
60	Gallons / Minute	Average
61	BTU	Sum
62	THERMS	Sum
63	Cubic Feet / Minute	Average
64	Cubic Feet / Second	Average
65	WM ²	Average
66	Relative Humidity	Average
67	MPH	Average
68	THI	Average
69	Gallons	Sum
70	Cubic Feet	Sum
71	Temperature Difference	Average
72	KVAR-OUT	Average

CODE	DESCRIPTION	AGGREGATION TECHNIQUE
73	KVA-OUT	Average
74	Knots	Average
75	Degrees	Average
76	CCF (Hundred cubic feet)	Sum
77	CF / Hour	Average
78	Pounds / Square inch	Average
79	Dollars	Sum
80	DECATHERMS	Sum
81	Pounds	Sum
82	Pounds / Hour	Average
83	MPOUNDS	Sum
84	MPOUNDS / Hour	Average
85	\$/KWH	Average
86	\$/MW	Average
87	\$/MWH	Average
88	\$/Hour	Average
89	Volt Hours	Sum
90	Individual Totalized Cubic Feet	Sum
91	Individual Totalized Btu	Sum
92	Pressure in MILLIBARS	Average
93	Visibility in Miles	Average
94	Cents per kWh	Average
99	Individual Totalized Gallons	Sum
100	MWH	Sum
102	Euros	Sum
103	Euros per MWH	Average
104	Euros MW	Average
105	GW	Average
106	TWH	Sum
107	Cubic Meters (M3)	Sum
108	Mega Joules per Cubic Meter (MJ/m3)	Average
109	Kilograms per Cubic Meter (Kg/m3)	Sum

CODE	DESCRIPTION	AGGREGATION TECHNIQUE
110	Cubic Meters per Hour (M3/H)	Average

UOM Compatibility

When merging cuts with different units-of-measure, cuts with specific differing UOMs can be combined while others can't. For example, you can't merge two cuts if the UOM of one cut is inches and the UOM of the other is dollars, because whichever UOM is assigned to the merged cut wouldn't apply to at least some of the data.

However, cuts with certain specific different UOMs can be combined. UOMs that can be combined are referred to as compatible UOMs, and are listed in the table below.

Compatible UOMs

01,31,32,51

02,52

03,33,34,53,56,57,58,59

04,36,54

05,35

07,37

08,09,38,39,55

11,41

12,42

13,43

17,47

18,48

22,72

23,73

61,69

69,99

70,90

These units of measures may have their intervals divided by the IPH for display:

Demand-Type UOMs

02, 05, 06, 07, 10, 22, 23, 24, 52, 72, 73,
105, 110

Overriding and/or Extending the Default UOM List

The above list represents the “default” list of UOMs used by Oracle Utilities applications. It is also possible to extend or override this “default” list if needed through the INTDCONFIG.CFG.XML configuration file. See **Overriding and/or Extending the Default UOM List** on page 3-6 for more information.

Enhanced Interval Data Storage

In standard interval data storage, interval data is always related to a Recorder-Channel combination, regardless of whether or not the data is actually related to an actual physical interval data recorder. Individual interval data readings, known as “cuts” are stored in the Channel Cut Header (or equivalent) table. Also, interval data cuts use a specific, non-flexible data structure, prohibiting the use of custom data elements on interval data records.

Enhanced (or “generic”) interval data storage removes the requirement that interval data be associated to a Recorder-Channel combination, and allowing interval data to be associated with any entity defined in the Oracle Utilities Data Repository. In addition, the enhanced/generic interval data allows for adding custom data elements to interval data cuts using meta-data.

This section describes enhanced/generic interval data storage in the Oracle Utilities Data Repository, including:

- **Parent Tables**
- **Category Tables**
- **Interval Data Tables**
- **Staging Tables**
- **Reporting Tables**
- **Interval Data Values**
- **Interval Data Status Codes**
- **Extended Status Codes**
- **Units of Measure**

Parent Tables

Parent tables are the tables that define the entities to which interval data is to be associated. These tables serve the same role that the Channel table serves in standard interval data storage. Any entity in the Oracle Utilities Data Repository can be used as a parent of enhanced/generic interval data. The identity columns of the example tables described below are noted in parentheses.

Example: MDM Meter

In the Oracle Utilities Meter Data Management application, the parent table for interval data is the MDM Meter (LSMDMTRDATACHANNEL) table. Records in this table are defined by the following:

- Meter ID (identity)
- Expected Unit of Measure (identity)
- Channel ID (identity)
- Service Point
- End Use Code
- Usage Type Code
- Related Channel
- Note

- Associated KVARH Data Channel

Example: Weather Station

One possible example of a parent table might be a Weather Station table, that defines specific weather stations that provide temperature data. Records in this table are defined by the following:

- Name (identity)
- Jurisdiction (identity)

Category Tables

Category tables are optional tables used as a means of further defining individual cuts of interval data. For example, a category might be used to define the status of a cut (raw, validated, or final). Category tables can be used as lookup tables on the interval data tables that store the individual interval data cuts.

Example: Usage Category (MDM)

In the Oracle Utilities Meter Data Management application, the category table for interval data is the Usage Category (LSMDUSAGECATEGORY) table. Records in this table are defined by the following:

- Category Code:
- Description

Example: Interval Data Category

A possible example category table might be simply called “Interval Data Category” which contains user-defined categories for interval data cuts (this table would be very similar to the Usage Category table). Records in this table might be defined by the following:

- Category
- Description

Interval Data Tables

Interval data tables are the tables that store the individual interval data cuts. These are the equivalent of the Channel Cut Header table used in standard interval data storage. Interval data tables are child tables of Parent tables (described above), in the same way that the Channel Cut Header table is a child of the Channel table in standard interval data storage. In addition, category tables can be used as optional lookup tables on interval data tables.

Parents and Categories

Interval data tables require a foreign key to the corresponding parent table. If a category is used, the interval data table must include a lookup to the category table.

Required Columns

In addition to columns defining the parent and optional category, there is a set of columns which are required on all interval data tables. These are attributes which define basic characteristics of the cut, such as the cut's Start Time, Stop Time, Seconds-Per-Interval, Unit-of-Measure, etc. In the table below, the Interval Data Attribute column lists the identifier used in the Oracle Utilities Rules Language when retrieving or defining the value of the column. The following columns are required on all enhanced/generic interval data tables:

Column Name	Interval Data Attribute	Description
INTERVALCOUNT	COUNT	The number of intervals in the cut

Column Name	Interval Data Attribute	Description
DSTPARTICIPANT	DSTPARTICIPANT	The cut's DST Participant flag
ORIGIN	ORIGIN	The cut's Origin flag
SPI	SPI	The number of seconds-per-interval
STARTTIME	STARTTIME	The cut's start time
STOPTIME	STOPTIME	The cut's stop time
CHNLCUTTIMESTAMP	TIMESTAMP	The cut's timestamp (when the cut was imported into the database)
TZSTDNAME	TZSTDNAME	The cut's Time Zone Standard Name
UOMCODE	UOM	The cut's unit-of-measure code
VALUECODES	N/A	Stores the individual interval values and status codes (binary format)

Optional Columns

In addition to the above required columns, enhanced/generic interval data tables can also contain other columns used in standard interval data storage, as well as columns based on calculated interval data attributes, such as TOTAL, ENERGY, etc. In the table below, the Interval Data Attribute column lists the identifier used in the Oracle Utilities Rules Language when retrieving or defining the value of the column. Optional columns based on columns used in standard interval data storage (such as CHANNELNUM, DESCRIPTOR, etc.) must be named **exactly the same as they are in the Channel Cut Header table**. Optional columns based on interval attributes must be named **exactly the same as the identifier used in the Oracle Utilities Rules Language** (listed in the Interval Data Attribute column). The following columns are optional on enhanced/generic interval data tables:

Column Name	Interval Data Attribute
ABS_MAXDATE	ABS_MAXDATE
ABS_MAXIMUM	ABS_MAXIMUM
AVERAGE	AVERAGE
AVERAGE_NZ	AVERAGE_NZ
CHANNELNUM*	CHANNEL
COUNT_NZ	COUNT_NZ
DC_FLOW	DC_FLOW
DELETE_FLAG	DELETE_FLAG
DELETEFLAG*	DELETE_FLAG_CHAR
DESCRIPTOR*	DESCRIPTOR

Column Name	Interval Data Attribute
DST_OFFSET	DST_OFFSET
DST_START	DST_START
DST_STOP	DST_STOP
DSTENERGY	DSTENERGY
DSTTOTAL	DSTTOTAL
DSTxxx	DSTxxx
EDIT_FLAG	EDIT_FLAG
EDITED*	EDIT_FLAG_CHAR
END_TIME	END_TIME
ENERGY	ENERGY
EXTERNAL_VAL_FLAG	EXTERNAL_VAL_FLAG
EXTERNALVALID*	EXTERNAL_VAL_FLAG_CHAR
INDEX	INDEX
INDEX_START	INDEX_START
INDEX_STATUS	INDEX_STATUS
INDEX_STOP	INDEX_STOP
INTERNAL_VAL_FLAG	INTERNAL_VAL_FLAG
INTERNAL_VAL_FLAG_CHAR	INTERNAL_VAL_FLAG_CHAR
IPH*	IPH
KWMAX*	KW_MAXIMUM
LOADFACTOR*	LF
MASK_FLAG	MASK_FLAG
MAXTIME*	MAXDATE
MAXIMUM*	MAXIMUM, MAX, >=
MAXINDEX	MAXINDEX
MERGE_FLAG	MERGE_FLAG
MERGE_FLAG_CHAR	MERGE_FLAG_CHAR
METERMULTIPLIER*	METER_MULT
METEROFFSET*	METER_OFFSET
MINTIME*	MINDATE
MINIMUM*	MINIMUM, MIN, <=
MINIMUMNZ*	MINIMUM_NZ, MIN_NZ
PULSEMULTIPLIER*	MULTIPLIER

Column Name	Interval Data Attribute
NEG_VALUES_FLAG	NEG_VALUES_FLAG
COUNTNON9VAL*	NON_9_VALUE
PULSEOFFSET*	OFFSET
POPULATION*	POPULATION
READING_VALUE	READING_VALUE
SPRING_DST	SPRING_DST
START_OFFSET_FLAG	START_OFFSET_FLAG
STARTREADING*	START_READING
STOPREADING*	STOP_READING
TIMEZONE*	TIMEZONES
TIMEZONES_FLAG	TIMEZONES_FLAG
TOTAL	TOTAL, TOTAL_VALUE
FLAGE*	VAL_FLAG_E
FLAGI*	VAL_FLAG_I
FLAGN*	VAL_FLAG_N
FLAGO*	VAL_FLAG_O
WEIGHT*	WEIGHT

* Columns are present on Channel Cut Header table in standard interval data storage.

See the **INTDVALUE Function** on page 9-79 in the *Oracle Utilities Rules Language Reference Guide* for descriptions of the above columns/attributes.

Custom Columns

In addition to the required columns, and optional columns used in standard interval data storage, enhanced/generic interval data tables can also include custom columns based on implementation-specific requirements. Custom columns are defined in meta-data in the same manner as custom columns on other tables in the Oracle Utilities Data Repository.

Example: Meter Data Channel Cut (MDM)

In the Oracle Utilities Meter Data Management application, the interval data table is the Meter Data Channel Cut (LSMDMTRDATA CUT) table. The MDM Meter table is the parent of this table, and the Usage Category table is a lookup on this table. Records in this table are defined by the following:

- Meter Data Channel (Parent - foreign key to the MDM Meter table)
- Usage Category (Category - lookup from the Usage Category table)
- All required columns (see **Required Columns** on page 11-13)
- All optional columns (see **Optional Columns** on page 11-14)
- Additional columns used by the Oracle Utilities Meter Data Management application

Example: Weather Data

A possible example of an interval data table might be a Weather Data table, that stores temperature data from specific weather stations (defined in the Weather Station table). The Weather Station table would be the parent of this table, and the Interval Data Category table could be used as lookup on this table. Records in this table are defined by the following:

- Weather Station (Parent - foreign key to Weather Station record)
- Interval Data Category (Category - lookup from the Interval Data Category table)
- All required columns (see **Required Columns** on page 11-13)

Staging Tables

Staging tables are optional tables used as a means of temporary storage interval data. Staging tables might be used to when data is initially imported into the database piece-meal, or in cases where the data is imported over an extended period of time. Once data is in a staging table, it can be moved into the appropriate enhanced interval data table using Rules Language.

Staging tables are typically associated to a specific enhanced/generic interval data table, and should contain all the Required, Optional, and Custom columns of the corresponding Header table.

Staging tables are unique in that they can store only a fixed number of intervals. For each interval to be stored, the staging table has a pairing of Value and Status columns, which store the interval value and status code for each interval. For example, if a staging table was going to be used to store up to 1 day of 15-minute data, it might have 100 Value/Status column pairs (1 for each of 96 intervals, plus 4 to accommodate the extra DST hour in the Fall).

Note: It is an error if you attempt to save an interval data handle to a staging table that does not have sufficient number of Value/Status column pairs.

Example: Weather Data Staging

A possible example of a staging table might be a Weather Data Staging table, that temporarily stores up to a year of hourly temperature data from specific weather stations (defined in the Weather Station table). Records in this table are defined by the following:

- Weather Station
- All required columns (see **Required Columns** on page 11-13)
- 9000 Value/Status columns pairs

Reporting Tables

Reporting tables are optional tables used as a means of temporarily storing interval data for reporting and/or data export.

Reporting tables are associated to a specific enhanced/generic interval data table, and should contain all the Required, Optional, and Custom columns of the corresponding Header table. Reporting tables have a child table in which individual interval records are stored. Each interval record contains the interval index, time, interval value, and status code.

Example: Weather Data Reporting

A possible example of a reporting table might be a Weather Data Reporting table, that temporarily stores temperature data from specific weather stations (defined in the Weather Station table) for reporting purposes. Records in this table are defined by the following:

- Weather Station
- All required columns (see **Required Columns** on page 11-13)

Interval Data Values

The VALUECODES column of enhanced interval data tables contains interval data values, interval data status codes, and (optionally) extended status codes in a binary format. The number (or count) of intervals, status code, extended status code triplets in this column is stored in the Interval Count column of the table. The space designated for this binary data is determined by the following calculation:

$$(T + 1) * (13 \text{ bytes}) = \text{length of binary column}$$

where:

- T = Total intervals (from INTERVALCOUNT column)

Each interval requires 13 bytes because each consists of an 8 byte double for the interval data value, a 1 byte ASCII character for the interval data status code, and a 4 byte integer for the extended status code.

Interval Values

Interval data values are stored contiguously. The first interval data value is contained in bytes 1 - 8, the second interval data value is contained in bytes 9 - 16, and the n th interval data value is contained in bytes $((8 * n) - 7)$ through bytes $(8 * n)$. After the last interval data value are 8 bytes of unused space.

Interval Status Codes

Interval data status codes are stored contiguously following these 8 bytes. The first interval data status is stored in byte $((8 * (T + 1)) + 1)$, the second interval data status is in byte $((8 * (T + 1)) + 2)$, and the n th interval data status is stored in byte $((8 * (T + 1)) + n)$.

After the last interval data status code there is one byte of unused space.

Extended Status Codes

Extended status codes are stored contiguously following the regular status codes. The first extended interval data status is stored in byte $((9 * (T + 1)) + 1 * 4)$, the second extended interval data status is in byte $((9 * (T + 1)) + 2 * 4)$, and the n th extended interval data status is stored in byte $((9 * (T + 1)) + n * 4)$.

After the last extended interval data status code there is 4 bytes of unused space.

Interval Data Status Codes

Enhanced/generic interval data uses the same status codes as standard interval data. See **Interval Data Status Codes** on page 11-7 for more information and list of standard status codes.

Extended Status Codes

Enhanced/generic interval data can use store extended status codes that provide additional details concerning the status of the overall interval data cut and of individual intervals. Extended status codes are represented by integer values (starting at 1) that define the status codes and the order in which the status codes exist. Integer values range from 1 through 32896, where each value is equal to twice the preceeding value (1, 2, 4, 8, 16, 32, etc.). Each integer value corresponds to a BIT, from 0 through 15. Integer values greater than 32768 (BIT 15) represent the sum of smaller integer values. The integer values that correspond to each bit are as follows:

Bit	Integer Value
0	1
1	2
2	4
3	8

Bit	Integer Value
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096
13	8192
14	16384
15	32768

There are two types of extended status codes: “CHANNEL” (channel-level status codes) and “INTERVAL” (interval-level status codes). The table below lists some pre-defined channel-level status codes:

BIT	Code	Description
0	NA	Not Used
1	AD	Added Interval (Data Correction)
2	RE	Replaced Interval (Data Correction)
3	ES	Estimated Interval (Data Correction)
4	OV	Pulse Overflow
5	HL	Data Out Of Limits
6	XC	Excluded Data
7	PY	Parity Error
8	TY	Energy Type (Register Changed)
9	LR	Alarm/Error
10	DI	Harmonic Distortion
11	IN	Point-to-Point Linear Interpolation
12	CK	Check Estimation
13	NA	Not Used
14	NA	Not Used
15	NA	Not Used

The table below lists some pre-defined interval-level status codes:

BIT	Code	Description
0	PO	Power Outage
1	SI	Short Interval (False For Mag Tape)
2	LI	Long Interval (Missing For Mag Tape)
3	CR	CRC Checksum Error
4	RA	RAM Checksum Error
5	RO	ROM Checksum Error
6	LA	Data Missing
7	CL	Hardware Clock Error
8	BR	Memory Reset
9	WT	Watchdog Timeout
10	TR	Time Reset
11	TM	Test Mode
12	LC	Load Control
13	NA	Not Used
14	NA	Not Used
15	NA	Not Used

To derive the actual status codes from the integer value, find the integer value on the table above, and locate the corresponding BIT (or BITs) on the list of channel- or interval-level status codes. For example, using the same codes above, the channel status code for a reading with a CHNSTATUS (channel status code) of “128” (BIT 7) would be displayed as “PY” (code), “Parity Error (description).

Multiple extended status codes can be specified in a cut by an integer that is equal to the sum of the integer values that correspond to the individual status codes. For example, to specify interval status codes “PO” (1, BIT 0) and “SI” (2, BIT 1), you would set this equal to “3”. To specify interval status codes “PO” (1, BIT 0), “SI” (2, BIT 1), and “CR” (8, BIT 3), you would set this equal to “11”.

Extended Status Codes can be defined in either the Interval Extended Status Code table in the Oracle Utilities Data Repository or in the EXTENDEDSTATUSES.CFG.XML configuration file. If the EXTENDEDSTATUSES.CFG.XML file is present in the C:\LODESTAr\CFG folder, it overrides settings in the Interval Extended Status Code table.

Note: The Interval Extended Status Code table is only available with Oracle Utilities Meter Data Management.

Units of Measure

Enhanced/generic interval data uses the same units of measure as standard interval data. See **Units of Measure** on page 11-7 for more information and list of standard status codes.

Interval Data Access

Interval data stored in the Oracle Utilities Data Repository can be accessed by users in a number of ways. This section outlines the available methods of accessing interval data using the Oracle Utilities Energy Information Platform, including:

- **Interval Data Viewer and Interval Data Manager**
- **Oracle Utilities Rules Language**
- **Interval Data Interface**

Interval Data Viewer and Interval Data Manager

The Interval Data Viewer component of the Energy Information Platform provides users with a tool to search and view interval data stored in either standard or enhanced interval data tables. See **Chapter 6: Searching and Viewing Usage and Interval Data** in the *Oracle Utilities Energy Information Platform User's Guide* for more information about viewing interval data using the Interval Data Viewer.

Interval Data Manager allows users to edit and manage interval data cuts. See **Chapter 9: Working with Interval Data Manager** in the *Oracle Utilities Energy Information Platform User's Guide* for more information about viewing interval data using the Interval Data Manager.

Oracle Utilities Rules Language

Interval data can also be accessed via Oracle Utilities Rules Language for use with other Oracle Utilities applications such as Oracle Utilities Billing Component, Oracle Utilities Quotations Management, or Oracle Utilities Meter Data Management. See **Chapter 7: Working with Interval Data** in the *Oracle Utilities Rules Language User's Guide* for details concerning accessing interval data via Oracle Utilities Rules Language. See **Chapter 9: Interval Data Function Descriptions** in the *Oracle Utilities Rules Language Reference Guide* for descriptions of interval data Rules Language functions.

Interval Data Interface

Interval data can also be accessed programmatically via the Interval Data interfaces (LSINTD.DLL). See **Chapter 20: Energy Information Platform Interval Data Interfaces** for more information about using the interval data interface.

Interval Data Versioning

The Interval data versioning feature maintains previous “versions” of interval data cuts as they change over time, either due to editing, or when replaced by more recent data. With versioning enabled, whenever an interval data cut is imported into the Oracle Utilities Data Repository that would overwrite an existing cut, the existing cut is first copied into a “version” table.

Note: Interval data versioning is supported for relational interval data sources only. It is **NOT** available for use with Btrieve (Pervasive.SQL) interval data sources.

Interval Data Version Tables

Interval data version tables are tables similar in structure to the Channel Cut Header table (or equivalent) that store versioned interval data cuts. In addition to the Channel Cut Header columns, interval data version tables have two additional columns:

- **Version Sequence:** A number that indicates the order in which the version of the cut was created. The lowest Version Sequence number for a given cut is the first version of that cut. A versioned cut with a higher Version Sequence is a more recent version of the cut.
- **Version Timestamp:** The date and time when the versioned cut was created.

Interval data version tables are associated to interval data sources in the Interval Data Sources table. See **Interval Data Sources** on page 3-12 for more information about this table.

The Oracle Utilities Data Repository database schema includes a version table that corresponds to the Channel Cut Header table. This table is called the Channel Cut Header Version table (LSCHVERSION).

Setup and Configuration

Interval data versioning is enabled by specifying a Version table for a relational interval data source in the Interval Data Sources table. See **Interval Data Sources** on page 3-12 for more information about setting up records in this table.

Note: Interval data versioning is enabled for the specific interval data table in the Oracle Utilities Data Repository specified in the Interval Data Sources record, NOT for the interval data source. This means that if more than one Interval Data Source record is based on the same interval data table (such as LSCHANNELCUTHEADER), and interval data versioning is enabled for any one of them, it is also enabled for ALL of them.

Note: Interval Data Versioning and Auditing (Edit Trails in IDM) cannot be enabled at the same time for the same data source.

Accessing Versioned Interval Data Via Oracle Utilities Rules Language

Versioned interval data can be accessed in the Oracle Utilities Rules Language via the INTDLOADVERSION function. Parameters for this function include:

- **Version Table Name:** The name of the version table where the versioned cut is stored
- **Recorder,Channel:** The Recorder ID and Channel Number of the versioned cut
- **Cut Start Time:** The start time of the versioned cut
- **Version Sequence (optional):** The Version Sequence of the versioned cut

Example:

Load a versioned cut for recorder,channel '1700,1' with a start time of January 1, 2004, and version sequence of 3.

```
VER_HNDL = INTDLOADVERSION(LSCHVERSION, '1700,1', '01/01/2004', 3);
```

See **INTDLOADVERSION Function** on page 9-50 in the *Oracle Utilities Rules Language Reference Guide* for more information about using this function.

Viewing Versioned Interval Data

Users can search and view versioned interval data using the Interval Data Viewer from the Oracle Utilities Energy Information Platform user interface. To access Interval Data Viewer, select **Usage->Interval Data** from the Left Menu. Versioned Data is accessed from the Cuts tab of the Interval Data Viewer.

See **Searching Interval Data** on page 6-3 and **Viewing Cut Versions** on page 6-13 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about searching and viewing versioned interval data.

Energy Information Platform Report Monitor

The Energy Information Platform Report Monitor is a Windows service that manages report generation and processes initiated by Oracle Utilities web-based products, including the Oracle Utilities Energy Information Platform, Oracle Utilities Quotations Management, Oracle Utilities Billing Component, Oracle Utilities Load Profiling and Settlement, Oracle Utilities Portfolio Management, and Oracle Utilities Transaction Management.

This service monitors the Report Instance table in the Oracle Utilities Data Repository (see **Energy Information Platform Reporting Framework Database Tables** on page 13-6) for reports to be run. The specific data source(s) monitored by the Report Monitor are specified in the LSREPORTMONITOR.CFG.XML file. See **LSREPORTMONITOR.CFG.XML** on page 2-35 for more information about this file.

See **How the Energy Information Platform Reporting Framework Works** on page 13-4 for more information about the report monitor service's role in the Reporting Framework.

Energy Information Platform Report Scheduler

The Energy Information Platform Report Scheduler is a Windows service that allows for scheduling (and scheduled recurrence) of reports and processes initiated by Oracle Utilities web-based products, including the Oracle Utilities Energy Information Platform, Oracle Utilities Quotations Management, Oracle Utilities Billing Component, Oracle Utilities Load Profiling and Settlement, and Oracle Utilities Transaction Management.

The specific data source(s) monitored by the Oracle Utilities Scheduler are specified in the LSSCHDLRR.CFG.XML file. See **LSSCHDLR.CFG.XML** on page 2-41 for more information about this file.

Note: When configuring the Report Scheduler, only **one** (1) Scheduler service should be configured to monitor a single data source. That is, if installing and configuring the Oracle Utilities Scheduler on more than one web server, multiple scheduler services should NOT monitor the same data source.

Energy Information Platform Logging Framework

The Energy Information Platform Logging Framework is a service that manages the creation of logging and error messages for Oracle Utilities applications, including Oracle Utilities web-based products such as the Oracle Utilities Energy Information Platform, Oracle Utilities Quotations Management, Oracle Utilities Billing Component, Oracle Utilities Load Profiling and Settlement, and Oracle Utilities Transaction Management, as well as client/server applications such as Data Manager, Oracle Utilities Billing Component, and Oracle Utilities Rate Management.

The specifics of how the Logging Framework creates logging and error messages are specified in the LSLOGGER.CFG.XML file. See **LSLOGGER.CFG.XML** on page 2-27 for more information about this file.

This section includes:

- **Logging Framework Output File Format**
- **Default Behavior**
- **Merging Tracing and Logging Files**
- **Tracing Levels**

Logging Framework Output File Format

The log file created by the Logging Framework has two parts: the header and the logging data.

Header Format

The header explains the logging data, and describes all fields in it. An example of header is bellow:

```
#Fields:
# 1) Datetime with thousands of second
# 2) Message type
# 3) Process name
# 4) Process ID
# 5) Thread ID
# 6) Message code
# 7) Message source (Component ID or other event source)
# 8) Source code file name
# 9) Source code line number
# 10) Message body
```

Data Format

Every message is represented by 9 service fields in one line and message body field, which could contain carriage return characters. There are two modes for service fields:

- Fixed widths
- Delimited

The width values for "fixed widths" mode:

Field name	Width
Date time	23
Message type	5
Process name	20
Process ID	6
Thread ID	6

Field name	Width
Message code	10
Message source	50
Source code file name	40
Source code line number	5

A sample log file appears below

Fixed widths mode:

```

2002-10-16 16:15:29.435 ERROR dllhost.exe          002608 001976 0x80040E14
Microsoft OLE DB Provider for ODBC Drivers
      clsdb.cpp                                     245 [Oracle][ODBC][Ora]ORA-
00904: invalid column name
2002-10-16 16:15:11.560 TRACE dllhost.exe          002608 001976 0x00050100 LSRDB
      crdbsource.cpp                               164 Prepared command:
sql='SELECT * FROM PWRLINE.TYPEMETADATA', params=''
2002-10-16 16:15:11.760 TRACE dllhost.exe          002608 001976 0x00050100 LSRDB
      crdbsource.cpp                               164 Prepared command:
sql='SELECT * FROM PWRLINE.DOMAINMETADATA', params=''
2002-10-16 16:15:11.960 TRACE dllhost.exe          002608 001976 0x00050100 LSRDB
      crdbsource.cpp                               164 Prepared command:
sql='SELECT * FROM PWRLINE.COLUMNMETADATA', params=''

```

Delimited mode:

```

2002-10-16 16:15:29.435|ERROR|dllhost.exe|002608|001976|0x80040E14|Microsoft
OLE DB Provider for ODBC Drivers|clsdb.cpp|245|[Oracle][ODBC][Ora]ORA-00904:
invalid column name
2002-10-16
16:15:11.560|TRACE|dllhost.exe|002608|001976|0x00050100|LSRDB|crdbsource.cpp|
164|Prepared command: sql='SELECT * FROM PWRLINE.TYPEMETADATA', params=''
2002-10-16
16:15:11.760|TRACE|dllhost.exe|002608|001976|0x00050100|LSRDB|crdbsource.cpp|
164|Prepared command: sql='SELECT * FROM PWRLINE.DOMAINMETADATA', params=''
2002-10-1616:15:11.960|TRACE|dllhost.exe |002608|001976|0x00050100|LSRDB
| crdbsource.cpp| 164|Prepared command: sql='SELECT * FROM
PWRLINE.COLUMNMETADATA', params=''

```

Default Behavior

By default, after installation, a logging configuration file does not exist. In this case, one of the following two “default” settings are used:

Type 1 Applications:

```

<lslogger>
  <file overwrite='NO' path='lodestar.log' />
</lslogger>

```

Type 2 Applications:

```

<lslogger>
  <file overwrite='YES' mode='PLAIN' path='ApplicationName.log' />
</lslogger>

```

where

- **ApplicationName** is the name of the executable or dll.

The default behavior of the logging framework depends on initial default settings of specific applications. For example, many Oracle Utilities console applications support a "-lcfg" switch to override default behavior and use a specified configuration file. In the lists below, those applications marked with an asterisk (*) do NOT support the "-lcfg" switch.

Type 1 applications include the following:

- Balance Accounts (BALACCT.EXE)
- Balance Journal (BALJRNLE.EXE)
- Batch Automatic Payments (BAUTOPAY.EXE)
- Pending Refunds (PENDRFND.EXE)
- Process Payment (PROCPMNT.EXE)
- Process Refunds (PROCRFND.EXE)
- Process Transactions (PROCTRNS.EXE)
- Update Refunds (UPDTRFND.EXE)
- Update General Ledger (UPGNLDGR.EXE)
- Update Subledger (UPSBLDGR.EXE)
- Process Voided Refunds (VOIDRFND.EXE)
- Report Monitor (LSREPORTMONITOR.EXE)*
- Report Scheduler (LSSCHDLR.EXE)*
- Workflow Engine (LSWFENG.EXE)*
- Transaction Management Broker (LTMHBROKER.EXE)*
- Secure Initialization (LSSECUREINIT.EXE)*
- Encrypt CFG Files (ENCRYPTCFG.EXE)
- Crystal Reports Report Generator (CRGENREPORT.EXE)*

Type 2 applications include the following:

- Data Manager (DATAMGR.EXE)*
- Oracle Utilities Load Profiling and Settlement Trial Settlement (LPSSBX.EXE)*
- Oracle Utilities Billing Component C/S (PLBX.EXE)*
- Oracle Utilities Rate Management (RATEXPRT.EXE)*
- Oracle Utilities Revenue Management (REVXPRT.EXE)*
- Autobill (AUTOBILL.EXE)
- Approval Required Billing (APRVREQ.EXE)
- Bill Correction (BILLCORR.EXE)
- Current/Final Billing (CURFINAL.EXE)
- Mass Billing (MASSBILL.EXE)*
- Automated Profiler (AUTOPROF.EXE)
- Export Customer/Account (EXPCA.EXE)
- Export Table (EXPTBL.EXE)
- Interval Data Export (INTDEXP.EXE)
- Interval Data Import (INTDIMPEX)
- List Import/Export (LSTIMEXP.EXE)
- List Refresh (LSTRFRSH.EXE)
- Meter Data Management System Export (MDMSEXP.EXE)*

- Meta-Data Validation (METAVAL.EXE)*
- Customer Data Import (PLIMPORT.EXE)
- RCATA to Relational Database (RCTORDBM.EXE)*
- Rate Form Import/Export (RFIMPEXP.EXE)
- Run Rate Schedule (RUNRS.EXE)
- Oracle Utilities Rules Language Engine (LSRLENG.EXE)*
- Import Manager (LSIMPORT.EXE)*
- Report Manager (LSREPORT.EXE)*
- Run Report (RUNREPORT.EXE)

Merging Tracing and Logging Files

The LogMerge.exe console utility can be used to merge the contents of multiple trace or log files into a single file. The resulting file contains all the tracing or logging messages from the merged files, sorted in chronological order (oldest to newest).

The syntax for this program is as follows:

```
LogMerge <merge file> <source file 1> [<source file 2> <source file n>]
```

where:

Parameter	Description
<merge file>	Path and file name of the merged file to be created. Example: c:\lodestar\log\Trace_merge.log
<source file>	Paths and file names of the files to be merged (up to 4), separated by spaces. Example: c:\lodestar\log\Trace1.log Trace2.log Trace3.log Trace4.log

Tracing Levels

The Logging Framework can be configured to capture tracing events at two different tracing levels. Certain tracing events are captured at each level. The table below details the information captured in trace files including:

- **Area:** The functional area for which tracing is enabled
- **Event:** The event captured in the trace file
- **Level:** The tracing level (INFO or VERBOSE, as specified in the “minlevel” attribute of the LSLOGGER.CFG..XML configuration file) at which the event is captured. For example, when capturing tracing information for Database Activity, the Metadata Loaded and Metadata Expired events are only captured if the “minlevel” attribute is set to “VERBOSE.”
- **Diagnose Information:** Diagnostic information provided in the trace file
- **Message:** The syntax for the trace message

Area	Event	Level	Diagnose Information	Message
All Executables	Start	Info	Command line	Command line: %1
		Verbose	System environment variables	System environment: %1
All consumers of configuration data	Configuration loaded	Verbose	Configuration data as file content, registry values.	Configuration loaded - Source: %1, Data:%2
Security	Session Begin	Info	User ID, Profile Id, Datasource Id, Session Id	Session Begin - Id:%1, User:%2, Profile:%3, DataSource:%4
		Verbose	Session Data	Session Begin - Data: %1
	Session Resume	Info	Session Id	Session Resume - Id:%1
	Session End	Info	Session Id	Session End - Id:%1
	Session datasource changed	Info	Session Id, Datasource Id	Session Datasource changed - Id:%1
	Session profile changed	Info	Session Id, Profile Id, Datasource Id	Session Profile changed - Id:%1, DataSource:%2
	Session profile language	Info	Session Id, Language Id	Session Language changed - Id:%1
	Session profile locale	Info	Session Id, Locale Id	Session Locale changed - Id:%1

Area	Event	Level	Diagnose Information	Message
Database activity	DB connection opened	Info	DB session id, Object Id, Connection String	DB Connection Opened - Session:%1, Object:%2, Connection String:%3
	DB connection closed	Info	DB session id, Object Id	DB Connection Closed - Session:%1, Object:%2
	Internal connection taken from pool	Info	Internal connection Id, DB connection's object id	Internal connection taken from pool - Id:%1, DB Connection Object:%2
	Internal connection returned to pool	Info	Internal connection Id, DB connection's object id	Internal connection released into pool - Id:%1, DB Connection Object:%2
	Pooled connection expired	Info	Internal connection Id, DB connection's object id	Pooled connection expired - Id:%1, DB Connection Object:%2
	Internal transactional connection taken	Info	Internal connection Id, Internal Transaction Id	Internal transactional connection taken - Id:%1, Internal Transaction Id:%2
	Internal transaction started	Info	Internal connection Id, Internal Transaction Id, Nested level	Internal transactional started - Id:%1, Internal Connection Id:%2, Nested level:%3
	Internal transaction committed	Info	Internal connection Id, Internal Transaction Id, Nested level	Internal transactional committed - Id:%1, Internal Connection Id:%2, Nested level:%3
	Internal transaction rolled back	Info	Internal connection Id, Internal Transaction Id, Nested level	Internal transactional rolled back - Id:%1, Internal Connection Id:%2, Nested level:%3
	DB transaction started	Info	DB transaction's object id, DB connection's object id	DB transaction started - Object:%1, DB Connection Object:%2

Area	Event	Level	Diagnose Information	Message
Database activity	DB transaction started	Info	DB transaction's object id, DB connection's object id	DB transaction committed - Object:%1, DB Connection Object:%2
	DB transaction started	Info	DB transaction's object id, DB connection's object id	DB transaction rolled back - Object:%1, DB Connection Object:%2
	Request executed	Info	SQL statement, Input parameters	Request executed - SQL: %1 Parameters: %2
	DML Request finished	Info	Affected records	DML request finished - Affected records: %1
	SP call finished	Info	Output parameters	SP call finished - Output Parameters: %1
	Metadata loaded	Verbose	Metadata Identity and structure	Metadata loaded - Identity: %1 Structure: %2
	Metadata expired	Verbose	Metadata Identity	Metadata expired - Identity: %1
File Upload	Start	Info	File Name	File upload start - Content:%1
	Finish	Info	File Size	File upload finished - Length:%1
Report framework	Template cached	Info	File Name, Report Id	Template cached - Report Name: %1, File:%2
	Cached template replaced	Info	File Name, Report Id	Template cache replaced - Report Name: %1, File:%2
	Report cached	Info	File Name, Report Id	Report cached - Report Name: %1, File:%2
	Report status changed	Info	Report Id, Status	Report status changed - Report GUID: %1, New Status:%2, Old Status:%3
	Report monitor started report generator	Info	Process Id, Number of currently running report generators	Starting report generator - Process Id: %1, Now running: %2
	Report monitor killed report generator	Info	Process Id	Killing report generator - Process Id: %1

Area	Event	Level	Diagnose Information	Message
Report framework	Report generator started	Verbose	Input parameters	Report generator started - Report Input Data: %1
	Report generator finished	Verbose	Report GUID	Report generator finished - Report GUID: %1
Rules Language engine	Rate form cached	Info	Rate form attributes	Rate form cached - Name: %1
	Rules language list cached	Info	List name	List cached - Id: %1 Global: %2
	Rate form started	Info	Rate form attributes, Input parameters	Rate form started - Name: %1, Input:%2
	Rate form started in debug mode	Info	Rate form attributes, Input parameters	Rate form started (debug) - Name: %1, Input:%2
	Rate form started	Verbose	User options, Global options, Billing options	Analysis Environment - Database configuration: %1, Default User Options: %2, Current User Options: %3, Billing Options: %4
	Rate form finished	Info	Rate form attributes	Rate form finished - Name: %1
	Rate form finished	Verbose	Profile, Statistics	Rate form profile: %1
	Rules language list cached	Info	List name	List cached - Id: %1

Area	Event	Level	Diagnose Information	Message
Client side SP procedures (LSDB4)	Loading/ caching of SP files	Info	File path	SP file processing: %1
	Stored procedure executed	Info	Input XML	SP call - Command:%1, Datasource: %2
	Stored procedure XSL transformation started	Verbose	Stored procedure's XSL content	XSL trasformation started - XSL:%1, Input:%2
	Stored procedure XSL transformation finished	Verbose	Transformation result	XSL trasformation finished - Result:%1

Area	Event	Level	Diagnose Information	Message
Adapter	Service started	Verbose	Service related configuration that located in database	Service started - Name:%1, Configuration:%2
	Processing of polled item started	Info	Polled item attributes (filename, JMS message id, payload UID and etc.)	File processing started - Name:%1
	Validation of polled item started	Verbose	Validation schema info	Validation started - Schemas:%1
	Validation of polled item finished	Verbose	Validation result	Validation finished - Errors:%1
	Transformation of polled item started	Verbose	Transformation script info	Transformation started - XSL:%1, Input:%2
	Transformation of polled item finished	Verbose	Result of transformation	Transformation finished - Result:%1
	Business Rule application started	Info	Business Rule attributes, parameters	File processing finished - Statistics:%1
	Business Rule application started	Verbose	Business Rule XML	Restart interval reached - Interval:%1
	Business Rule application finished	Verbose	Result XML	Business Rule started - Name:%1, Parameters:%2
	Processing of polled item finished	Info	Statistics gathered during processing.	Business Rule started - XML:%1
	Restart interval reached	Info	Interval value	Business Rule finished - Result:%1

Area	Event	Level	Diagnose Information	Message
MDM	Import: MDMUSAGE processing started	Verbose	MDMUSAGE element w/o child elements	Usage processing started - XML:%1
	Import: METERREAD processing started	Verbose	METERREAD element	Meter Read processing started - XML:%1
	Import: METEREVENT processing started	Verbose	METEREVENT element	Meter Event processing started - XML:%1
	Critical/ Required Validation done	Verbose	Validation type	Critical Validation done - Type: %1
	Optional Validation started	Verbose	Validation Group XML Structure	Optional Validation started - XML: %1
	Optional Validation finished	Info	Validation Group, Result	Optional Validation finished - Result: %1
	Estimation started	Info	Estimation type and parameters	Estimation started - Type: %1, Parameters: %2
	Estimation started	Verbose	Estimation rules	Estimation started - Rules: %1
	Estimation finished	Info	Estimation result	Estimation finished - Result: %1

Database Auditing

Database auditing is the process of tracking changes made to database records (also known as database events) in the Oracle Utilities Data Repository, including inserting new records, and updating and/or deleting existing records. Database auditing allows system administrators and database administrators to track changes made to database records, as well as identify the specifics of these changes. This section describes database auditing in the Oracle Utilities Data Repository, including:

- **Types of Database Auditing**
- **Storing Auditing Information**
- **Installing Auditing**
- **Auditing Configuration and Administration**
- **Auditing Reports**

Types of Database Auditing

There are two types of Oracle Utilities Data Repository database auditing: Basic auditing and Full auditing.

Basic Auditing

Basic auditing tracks insert and update database events by updating the LSUSER (User ID) and LSTIME (Timestamp) columns whenever a record is inserted or updated. This allows system administrators and database administrators to easily track which user was the last to update records in the Oracle Utilities Data Repository.

Full Auditing

Full auditing tracks database events by storing details of insert, update, and delete events in the LSDBAUDIT (LODESTAR Database Audit) table in addition to updating the LSUSER (User ID) and LSTIME (Timestamp) columns whenever a record is inserted or updated. Full auditing allows system administrators and database administrators to track which users made changes to database records, as well as identify the specifics of the changes made to the database.

Storing Auditing Information

Auditing information is stored in different places, depending on which type of auditing (Basic or Full) is enabled.

LSUSER and LSTIME

When using both Basic and Full auditing, the LSUSER and LSTIME columns are used to track the user who performed the database event and the time the event was performed. These columns are added to each table in the database when auditing is installed with the Oracle Utilities Data Repository.

Full Auditing - Database Auditing Table

When using Full auditing, details of database events are stored in the LSDBAUDIT (LODESTAR Database Audit) table. Records in this table contain the following:

- **UID:** unique identifier for the database event
- **EVENTTIME:** the time of the database event
- **EVENTTYPE:** the type of database event (I=Insert, U=Update, D=Delete)
- **USERID:** the user ID of the user who performed the event

- **TABLENAME:** the table in which the event took place
- **RECORDKEY:** the key of the record related to the event
- **OLDVALS:** the value(s) of the database record prior to the event
- **NEWVALS:** the value(s) of the database record after the event
- **OLDVALSCLOB:** the value(s) of CLOB columns in the database record prior to the event
- **NEWVALSCLOB:** the value(s) of CLOB columns in the database record after the event

Database Audit CLOBs Table

By default, Full auditing does not include auditing of changes made to columns that store CLOB (Character Large Object) data. However, auditing of these columns can be specifically enabled by administrators (see **Auditing Configuration and Administration** on page 11-39). The Database Audit CLOBs table defines the specific CLOB columns that should be audited. Records in this table contain the following:

- **UIDCOLUMN:** the UIDCOLUMN (from the Column Meta-Data table) of the CLOB column
- **ISAUDITABLE:** a flag that indicates if the CLOB column is auditable

Installing Auditing

Database auditing is installed when the Oracle Utilities Data Repository database schema is created (see **Chapter 3: Oracle Utilities Data Repository Schema Creation** in the *Oracle Utilities Energy Information Platform Installation Guide* for more information about creating the database schema).

A Windows Script File called **genAuditTrgs.wsf** is provided to generate the database triggers that provide the auditing functionality. This script is executed automatically (using default values) when the database schema is created. The syntax for this script is:

```
genAuditTrgs.wsf [/cmd:<batch_mode>] [/dsn:<dsn>] [/mode:<mode>]
[/tables:<table_list>] [/exclTables:<exclude_tables>]
[/stateFile:<file_name>]
```

Parameter	Description
<batch_mode>	Batch mode: 'yes' indicates to take arguments from command line; 'no'(default) indicates to take arguments arguments from user input

Parameter	Description
<dsn>:	<p>Database connection information for the Oracle Utilities Data Repository that contains the rate form record. This parameter is required and must be in one of the following formats:</p> <p>For Oracle databases:</p> <p>"Data Source=<data_source>;User ID=<user_id>;Password=<password>;LSProvider=ODP;"</p> <p>For Microsoft SQL databases:</p> <p>"Data Source=<address>;Initial Catalog=<SQL_database>;User Id=<user_id>;Password=<password>;MultipleActiveResultSets=True;LSProvider=MSSQL;"</p> <p>where:</p> <ul style="list-style-type: none"> • <data_source> is the Oracle TNS Name for the data source, from the TNS_NAMES.ora file (typically located in the \\<machine>\oracle\network\admin directory) • <user_id> is the user ID for the database connection • <password> is the password for the supplied user ID. • <address> is the IP address or Hostname of the MS SQL Server database server • <SQL_database> is the name of the MS SQL Server database
<mode>	Mode in which the script is executed: 0(default) - generate database triggers; 1 - Save status of database triggers to file
<table_list>	A comma-separated list of tables for which to generate database triggers. If omitted, database triggers will be generated for all tables defined in meta-data
<exclude_tables>	A comma-separated list of tables excluded when generating database triggers
<file_name>	The name of file that contains the state of triggers. Required when mode=1.

Examples:

```
genAuditTrgs.wsf
```

```
genAuditTrgs.wsf /cmd:yes /dsn:"DSN=SOMETHING;UID=USERID;PWD=PASSWORD"
```

```
genAuditTrgs.wsf /cmd:yes /dsn:"DSN=SOMETHING;UID=USERID;PWD=PASSWORD"
 /mode:1 /stateFile:auditTrgsState.xml
```

```
genAuditTrgs.wsf /cmd:yes /dsn:"DSN=SOMETHING;UID=USERID;PWD=PASSWORD"
 /stateFile:auditTrgsState.xml
```

```
genAuditTrgs.wsf /cmd:yes /dsn:"DSN=SOMETHING;UID=USERID;PWD=PASSWORD"
 /tables:"ACCOUNT,CUSTOMER,LIST"
```

```
genAuditTrgs.wsf /cmd:yes /dsn:"DSN=SOMETHING;UID=USERID;PWD=PASSWORD"
 /exclTables:LIST
```

Auditing Configuration and Administration

Once auditing is installed with the database schema, an administrator must configure auditing for the specific table(s) to be audited. Auditing configuration and administration is performed via the Security User Interface, and is configured for each data source defined in the Security Administration.

See **Configuration: Auditing** on page 7-29 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about auditing configuration.

Auditing Reports

The Auditing configuration screen also provides access to an Audit report that allows administrators to view and download/export a report that lists details regarding database changes. Changes are logged by column, meaning if a user changes values in three columns of a single record, the Audit Report will list three records, one for each column.

Each change listed on the Audit Report includes the following:

- **Time:** the time at which the change was made.
- **User:** The User ID of the user who made the change.
- **Event:** The type of change (U=Update, I=Insert, D=Delete).
- **Table:** The table in which the change was made.
- **Key:** The resolved identity of the record, displayed as a comma-separated list.
- **Column or Full:** The column in which the change was made (for Update events), or the full comma-separated record (for Insert or Delete events).
- **Value Before:** The value of the column before the change (Update events only).
- **Value After:** The value of the column after the change (Update events only).

CLOB columns are designated by a CLOB link. For Updates to CLOB columns, a CLOB link appears in both the **Value Before** and **Value After** columns. For Inserts and Deletes, a CLOB link appears within the comma-separated record.

See **Audit Reports** on page 7-31 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about audit reports.

Chapter 12

Energy Information Platform Security

This chapter describes the security features of the Oracle Utilities Energy Information Platform, including how to implement and set up security for Oracle Utilities web-based applications. This includes:

- An **Overview** of Oracle Utilities Energy Information Platform Security and the specific applications that can be secured
- **Security Concepts and Data**
- **Working with Function-Level Security**
- **Working with Record-Level Security**
- **The Security User Interface**
- **Setting Up Security**
- **Integrating Security with LDAP**
- **Single Sign-On**

Each of these topics is discussed in detail in this chapter.

Important Note

Except where specifically noted otherwise, the use of the term “data source” in this chapter refers to logical data sources, which consist of:
<CONNECTSTRING, QUALIFIER>.

Overview

Oracle Utilities Energy Information Platform security offers session-based application and data security for Oracle Utilities web-based applications. This includes secure access to specific applications, functions, and options of Oracle Utilities web-enabled products, as well as secure access to data in the Oracle Utilities Data Repository. Security can be implemented to provide secure access to the following:

- Global Functions (enabling/disabling general options and administrative functions common to all Oracle Utilities web applications)
- Menu Items (enabling/disabling specific applications as appropriate)
- Applications Functions (enabling/disabling specific functions within applications as appropriate)
- Fields/Controls (enabling/disabling specific fields/controls within functions as appropriate)
- Database tables and records (enabling/disabling access to specific tables and records)

How Security Works

At a high-level, Energy Information Platform security works in the following manner:

Users are assigned Data Sources and Groups.

- Data Sources are specific instances of the Oracle Utilities Data Repository.
- Groups represent a groups of users (each which might be a type of role the users assume as users of the applications). Groups determine the specific applications, functions, and options available to users, as well as the specific data users have access to.

When a user logs in to the Oracle Utilities Energy Information Platform, the Data Source and Group assigned to them determine the specific data they can access, the functions they can perform, and scope of those functions.

For more information, see **Security Concepts and Data** on page 12-3.

What Applications Are Secured?

Oracle Utilities Energy Information Platform security provides secure access to Oracle Utilities web-based applications, including the Oracle Utilities Energy Information Platform, Oracle Utilities Meter Data Management, Oracle Utilities Quotations Management, Oracle Utilities Billing Component, Oracle Utilities Load Profiling and Settlement, and Oracle Utilities Transaction Management.

Security Concepts and Data

Before setting up and implementing security, it's important that you understand a number of key concepts.

Security Modes

Oracle Utilities Energy Information Platform security can be run in one of two modes: Default mode, or LDAP mode.

Default Mode

Default mode is the default mode in which security is run. In this mode, all security information, including user authentication data (user IDs and passwords), is stored in the Security schema in Oracle Utilities Data Repository.

LDAP Mode

Lightweight Directory Access Protocol (LDAP) is a standard means of accessing a central directory service used to store user authentication information (and other directory resource information). In LDAP mode, user authentication data is stored in an external LDAP server instead of the Oracle Utilities Data Repository. All other security-related data, including groups, data sources, and sessions are still stored in the Security schema when in LDAP mode.

The use of LDAP mode with Security is specified in the LSSECURE.CFG file. See **LSSECURE.CFG.XML** on page 2-42 for more information about setting up this file.

See **Setting Up Security** on page 12-15 for more information about setting up Security to run in LDAP mode.

See **Integrating Security with LDAP** on page 12-23 for more information about integrating security with LDAP.

Types of Security

Oracle Utilities Energy Information Platform security provides for two primary types of security: function-level and record-level.

Function-level security

Function-level security is the means by which administrators grant privileges (or permissions) to groups and/or individual users. Privileges for tables (and Add, Update, and Delete operations on tables) in Data Navigator are defined for individual data sources using the Security Administration screens.

Record-level security

Record-level security is the means by which administrators grant access to specific records in specific tables (defined using the Security Administration screens) to groups and/or individual users. Record-level security applies to all screens in the Oracle Utilities Energy Information Platform and in all web-based products.

Record-level security allows for two types of security: direct and indirect.

- **Direct Security:** Allows for securing specific tables, such as the Account or Customer table.
- **Indirect Security:** Is based on the consequences (inherited or specified) of direct security.
 - **Inherited Indirect:** This is indirect security that flows down from a Parent to Child table. For example, securing an Account also secures all child records and descendants of that Account.

- **Specified Indirect:** This is indirect security that flows up from a Child to Parent table, where the Child table in question has been secured in some manner, either via Direct or Inherited Indirect security. In this case, access to a parent record is only allowed if the user has access to one or more child records of that parent. For example, a client might want to secure the Recorders table such that only recorders associated with a specific Account (or Accounts) are accessible. This could be done by directly securing the Account table (and thus indirectly securing the Channel History table), and then securing the Recorder table via the Channel and Channel History tables.

There are three ways to grant access to records in a secured table using record-level security:

1. All records in the table, or
2. Specific records in the table (known as Keys), or
3. Records in the table returned from a SQL query

In all three cases, access to records is defined for each individual group and/or user. This means that one group or user might have access to all records in a table, while another group or user might have access to only a specific set of records in that table.

Security Data

Data Sources

A Data Source defines a unique connect string/qualifier combination required to access an application database. Data Sources are defined by the following data:

Data Source ID: A unique ID for the data source.

Name: Name of the data source.

DBMS Type: The type of Database Management System. This can be either Oracle or SQL Server.

Connect String: The connect string used to connect to the data source.

Qualifier: A qualifier for the data source.

Is Default: A flag that indicates if the data source is the global default data source for the application.

Area Schema (Configuration - Data Access)

Each data source can optionally have an area schema that defines the specific tables within the data source that are to be secured, either directly or indirectly.

Permission Schema (Configuration - Navigator Access)

Each data source can optionally have a permissions schema that defines the specific tables that can be secured within the Data Navigator application. If a table is secured in this way, that table, and all its children and descendents, are no longer available to Data Navigator.

Determining the Default Data Source

When a user logs on to an application, the Security functionality determines the default data source for the session based on the following logical steps:

1. The system uses the default data source for the user. This is a User Data Source for the user that has the **Is Default** flag set to Yes.
2. Else, the system uses the first User Data Source (alphabetically) for the user.
3. Else, the system uses the Global Default Data Source. This is a Data Source that has the **Is Default flag** set to Yes.
4. Else, the system uses the first Data Source (alphabetically).

Groups

A group defines a set of permissions and/or access to specific records in secured tables.

Groups are Data Source specific: Specific permissions and/or data access for a group is defined for a specific data source, and can be defined differently for different data sources (though they must use the same schemas).

Feature Privileges

Feature privileges determine the specific functions available in an application or applications. Feature privileges are defined by an **application permissions schema file** (stored on the web server), and (optionally) a data source permissions schema (stored in the data base). Permission schemas contain a hierarchical structure (in XML format) that specifies the possible types of permissions that each function can have.

Different types of permissions are supported, including:

- **General:** Indicated by the existence of a function in the permissions structure.
- **Enabled:** Indicates that the function is enabled on the interface.
- **Read:** Indicates that the user/group has read access to data.
- **Write:** Indicates that the user/group has write access to data.
- **Execute:** Indicates that the user/group can execute a function.
- **Select:** Indicates that the user/group can select a function.
- **Update:** Indicates that the user/group can update data.
- **Insert:** Indicates that the user/group can insert records.
- **Delete:** Indicates that the user/group can delete data.

Within the permissions schema, specific permissions are attributes of a node representing a function.

Access to a function can be further restricted by maximum or minimum values, or one or more Areas.

Data Privileges

Data Privileges define specific areas of data that are available to a user or group of users. Data Privileges are defined by an **areas schema** stored in the data base. An area schema is a flat structure (in XML format) that specifies the specific tables that can be secured for a particular data source.

Users

Users are the individual users of the applications available through the Energy Information Platform. Users are defined by the following data:

- **User ID:** A unique ID for the user.
- **Password:** A password used when logging onto an application. Password-related requirements can be specified in the LSSECURE.CFG.XML configuration file.. See **LSSECURE.CFG.XML** on page 2-42 for more information.
- **Properties:** A set of properties associated with the user.

Profiles

Each user can have one or more profiles. Each profile can be granted access to a unique set of data sources and groups.

Personal Privileges

Users can have personal privileges, similar to those associated with Groups. Like Groups, users have Personal Privileges for Features (permissions) and for Data (areas).

Important Note

If running in LDAP mode, user authentication is performed using data in the LDAP server, not in the Oracle Utilities Data Repository.

Sessions

A Session is a series of one or more request from a single user. When a session is begun, all security information for that session is determined, including:

- Data Source
- User Properties
- Permissions
- Areas

Additional properties can be created during the session. Properties can be created or updated for just the single sessions for both the session and the user.

Sessions are configured to end automatically after a configurable period of inactivity. Users can end their own sessions by logging off, and administrators can end any session.

Working with Function-Level Security

Function-level security is the means by which administrators grant privileges (or permissions) to groups and/or individual users. Privileges can include access to menu options, screens, and features (defined by security schema files on the web server), as well as access to tables (and Insert, Update, and Delete operations on those tables) in Data Navigator (defined for individual data sources using the Security Administration screens).

Types of Function-Level Security

There are two types of function-level security:

- **Menu Option, Screen, and Feature Security**
- **Table and Column Security**

Menu Option, Screen, and Feature Security

Menu option, screen, and feature security refers to securing specific menu options, screens, and/or features, such as the Accounts or Customers options in the Entity Management menu.

Table and Column Security

Table security refers to securing specific tables (as well as Insert, Update, and Delete operations on those tables) and columns (and search, list, and edit functions on those columns) with Data Navigator.

Applying Function-Level Security

The steps involved in applying function-level security differ when applying security to menu options, screens and features than when applying security to Data Navigator tables.

- Applying function-level security to menu options, screens and features involves **Granting Feature Privileges**.
- Applying function-level security to a table (or tables) and related column(s) involves **Configuring Navigator Features**, and **Granting Feature Privileges**.

Configuring Navigator Features

Configuring Navigator Features is performed using the Security Administration user interface, and involves designating the table(s) and column(s) to be secured.

Tables and columns designated on this screen appear in the Feature Privileges tree. In addition:

- Insert, Update, and Delete privileges can be granted for each table.
- Search View, List View, Detail View, and Edit privileges can be granted for each column.

About Securing UID Columns

Use caution when securing UID columns on tables, as doing so can hide data based on a UID. For example, the UIDRATEFORM column on the Rate Code table is the source of the “Rate Form” column on the Search, List, and Edit screens for this table in Data Navigator. If the UIDRATEFORM column is secured, the “Rate Form” will not be available unless specific privileges for the Search View, List View, and Detail View are granted for that column.

Granting Feature Privileges

Granting Feature Privileges means to specify which features (including menu options, screens, features, and tables) are allowed by the user. This is performed using the Feature Privileges screen.

Access to features is defined for each individual group and/or user. This means that one group or user might have access to all features and tables (such as might be the case with a system administrator), while another group or user might have access to only a specific set of features and/or tables.

Working with Record-Level Security

As described earlier, record-level security is the means by which administrators grant access to specific records in specific tables (defined using the Security Administration screens) to groups and/or individual users. Record-level security applies to all screens in the Oracle Utilities Energy Information Platform and in all web-based products.

Types of Record-Level Security

Record-level security allows for two types of security:

- **Direct Security**
- **Indirect Security**

Direct Security

Direct security refers to securing specific tables, such as the Account or Customer table.

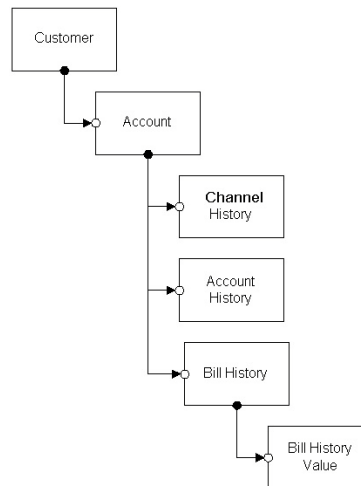
Indirect Security

Indirect security refers to the consequences of direct security. There are two types of indirect security: inherited and specified.

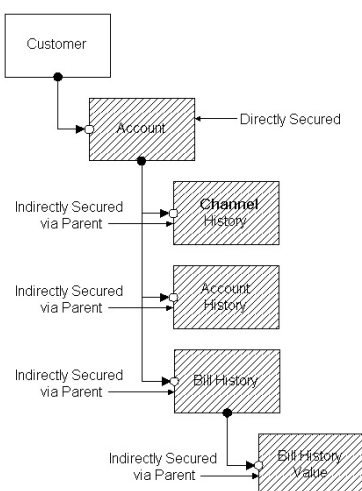
Inherited Indirect Security

Inherited indirect security is indirect security that flows down from a Parent to Child table. Inherited indirect security occurs automatically whenever a table is Directly secured.

Example: Securing the Account table also secures all children and descendants of the Account table. The following diagram displays the relationships between tables in a small portion of the Oracle Utilities Data Repository applicable to this example.



If the Account table is Directly secured, the children and descendants of the Account table are Indirectly secured via **inherited** indirect security.

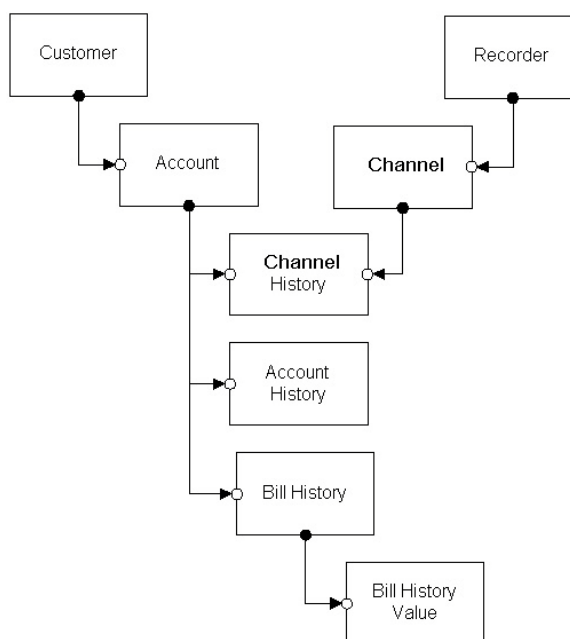


Specified Indirect Security

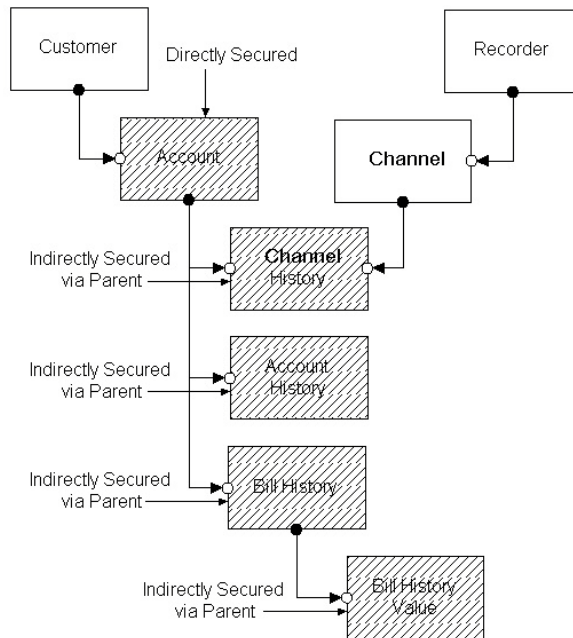
Specified indirect security is indirect security that flows up from a Child to Parent table, where the Child table in question has been secured in some manner, either via Direct or Inherited Indirect security. In this case, access to a parent record is only allowed if the user has access to one or more child records of that parent.

Example: You might want to secure the Recorders table such that only recorders associated with a specific Account (or Accounts) are accessible. This could be done by directly securing the Account table (and thus indirectly securing the Channel History table), and then using specified indirect security to secure the Recorder table via the Channel and Channel History tables.

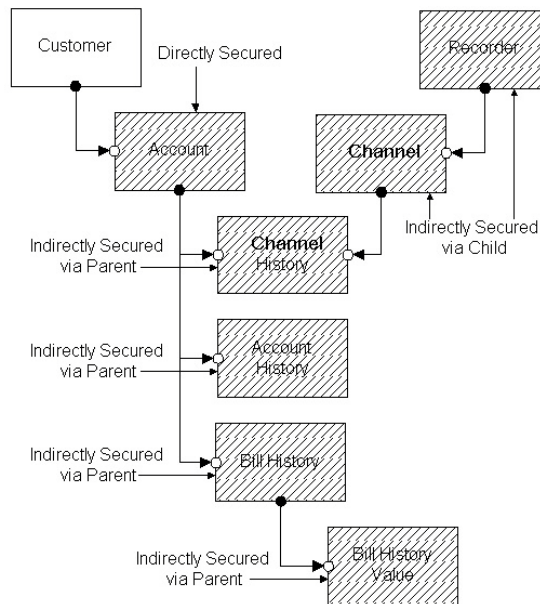
The following diagram displays the relationships between tables in a small portion of the Oracle Utilities Data Repository applicable to this example.



If the Account table is Directly secured, the children and descendants of the Account table are Indirectly secured via **inherited** indirect security.



The Channel and Recorder tables can then be indirectly secured via **specified** indirect security.



Applying Record-Level Security

There are two steps in applying record-level security to a table (or tables): Configuring Data Access, and Granting Data Privileges.

Configuring Data Access

Configuring Data Access for a table (or tables) is performed using the Security Administration user interface, and involves designating the table(s) to be secured as either Direct Tables (Direct Security) or Tables Secured by Child (Specified Indirect Security). See **Data Sources** on page 7-21 in the *Oracle Utilities Energy Information Platform User's Guide* for more information.

Granting Data Privileges

Granting Data Privileges for a table means to specify which records in that record the user and/or group can access. This is performed using the Data Privileges screen.

You only grant data privileges to **directly** secured tables. Indirectly secured tables are secured via either a Parent>Child or Child>Parent relationship. Once data privileges are granted to a directly secured table, inherited indirect security is applied automatically to all children and descendants of the secured table, and specified indirect security is applied to those tables so designated.

There are three ways to grant data privileges to records in a secured table using record-level security:

1. All records in the table, or
2. Records in the table returned from a SQL query
3. Specific records in the table (known as Keys), or

In all three cases, access to records is defined for each individual group and/or user. This means that one group or user might have access to all records in a table, while another group or user might have access to only a specific set of records in that table. See **Granting Data Privileges** on page 12-12 for more information.

Examples of Using Record-Level Security

The following examples describe how to configure data access for record-level security for a number of different business scenarios. Each example lists the specific table(s) to be secured via Direct and Indirect security to meet the criteria of the scenario. Note that these examples are limited to illustrating how to grant data access, and don't include granting data privileges.

Scenario: Secure Customer/Accounts and related Recorders via Account

Type of Security	Tables Secured
Direct	Customer
Indirect	Channel (Channel History) Recorder (Channel)

Scenario: Secure Customers/Accounts via Billing Cycle

Type of Security	Tables Secured
Direct	Billing Cycle
Indirect	Account (Account History) Customer (Account)

Scenario: Secure Customers/Accounts and related Recorders via Jurisdiction

Type of Security	Tables Secured
Direct	Jurisdiction
Indirect	Channel (Channel History) Recorder (Channel) Customer (Account)

Scenario: Secure Customers/Accounts and related Recorders via Operating Company

Type of Security	Tables Secured
Direct	Operating Company
Indirect	Channel (Channel History) Recorder (Channel) Customer (Account)

Limits on Record-level Security

Record-level security offers a powerful and flexible manner for restricting access to data in the Oracle Utilities Data Repository. However, there are some limitations associated with record-level security that need to be considered.

Performance Impact

Applying record-level security to a table (or tables) can have an impact on the performance of database searches on that table and its children and descendents (tables secured via implicit indirect security as a result of direct security). The factors that may impact performance of database searches includes:

- The number of directly secured tables
- The number of indirectly secured tables (via both implicit and specified indirect security)
- The number of parent-child links between a searched table and a secured table
- The number of keys specified for a directly secured table
- The number of conditions in the SQL "where" clause for a directly secured table
- The presence of the "Allow Null" attribute for a directly secured table
- The number of records in a secured table
- The number of records in a searched table
- The number of records in any table that is in the path of parent-child links between a searched table and a secured table (for instance, if the Account table is directly secured and you search the Bill history Value table, the number of records in the Bill History table impact the performance of the search)

As a general rule, where possible, each of these factors should be as small as possible. That is, when configuring record-level security, apply direct security and specified indirect security to as few tables as possible, and apply as few restrictions (keys, SQL conditions, etc.) as possible for the best performance results.

The Security User Interface

The Security Administration user interface is accessed via the Security Admin option under the Tools and Utilities Menu. Selecting that option opens the Security user interface in the Main Window.

See **Security Administration** on page 7-18 in the *Oracle Utilities Energy Information Platform User's Guide* for details on using the Security Administration user interface.

Setting Up Security

This section provides information useful when setting up Oracle Utilities Energy Information Platform Security, including:

- **Setting Up Security in Default Mode**
- **Setting Up Security in LDAP Mode**

Setting Up Security in Default Mode

Setting up Security in default mode involves the following steps.

1. **Set Up Security Configuration File**
2. **Encrypt Configuration File (optional)**
3. **Set Administration Login Password**
4. **Administrator Login**
5. **Set Up Security Data**
 - a. **Data Sources**
 - b. **Groups**
 - b. **Users**

Set Up Security Configuration File

The Security functionality is enabled by the presence of the LSSECURE.CFG.XML configuration file installed in the **C:\LODESTAR\CFG** directory.

When running security in default mode, this file contains two elements: the data source used by the Security functionality (required) and an optional session expire time (in minutes).

The data source specified is the data source that contains the security data (Data Sources, Groups, and Users). If not included, the expire time defaults to 60 minutes.

LSSECURE.CFG Example - Default Mode

```
<LSSECURE>
  <DATASOURCE>
    <NAME>PWRLINE</NAME>
    <CONNECTSTRING>Data Source=<data_source>;User
ID=<user_id>;Password=<password>;LSProvider=ODP;</CONNECTSTRING>
    <QUALIFIER>PWRLINE</QUALIFIER>
  </DATASOURCE>
  <EXPIRETIME>120</EXPIRETIME>
</LSSECURE>
```

Important

This file **MUST** be edited to specify the security data source prior to running Oracle Utilities applications.

See **LSSECURE.CFG.XML** on page 2-42 for more information about setting up this file.

Encrypt Configuration File (optional)

Before logging on to the Oracle Utilities application, the connect string in the LSSECURE.CFG.XML file (contained in the <CONNECTSTRING> element) can be encrypted using the EncryptCFG.exe command line program. This program must be run on the web/application server where the Oracle Utilities applications are installed. The syntax for this program is as follows:

```
EncryptCFG -f <file name> [-?]
```

where:

Parameter	Description
-f	Name of the configuration file to be encrypted. Example: -f lssecure.cfg.xml
-?	Used to display syntax.

See **Encrypting Configuration Files** on page 2-54 for more information about encrypting configuration files.

Set Administration Login Password

Before logging on to the Oracle Utilities application, the administrator password must be set using the LSSecureInit.exe command line program. This program must be run on the web/application server where the Oracle Utilities applications are installed. The syntax for this program is as follows:

```
LSSecureInit -d <set_pw>
```

where:

Parameter	Description
-d	<set_pw> prompts the user to set a password for administrator login (using the “admin” user ID) is to be set. The user is prompted to enter and re-enter the password.

Administrator Login

When logging into the Security user interface for the first time (before setting up any data sources, groups, or users), administrators should login as the default security administrator using the following User ID and Password:

User ID: “admin”

Password: *

The administrator password is set using LSSecureInit.exe command line program, as described in **Set Administration Login Password** above.

The default administrator is assigned feature privileges for a Data Source named **SECURITY INTITAL**, and has access to all the Security functions. This enables a first-time user of the Security user interface to set up the required data sources, groups, and users. Oracle recommends that you set up an alternate security administrator user that has access to all the Security functions for general use.

Important Note

Do not delete the “admin” user or the SECURITY INITIAL data source until and/or unless you have created an alternate security administrator user that has access to all the Security functions.

Set Up Security Data

Setting up Security involves setting up the following types of data:

Data Sources

First, set up the applicable data sources using the Data Sources tab of the Security user interface. Perform the following steps for each data source you need to set up:

1. Click **Add Data Source** on the Data Sources tab.
2. Enter the following information:
 - **ID:** An ID for the data source. This **MUST** be unique.
 - **Name:** A name for the data source.
 - **Type:** The type of data source. Select Oracle.
 - **Qualifier:** The database qualifier for the data source.
 - **Connect String:** The connect string for the data source. A connect string consists of the following elements, separated by semi-colons:

For Oracle databases:

```
<CONNECTSTRING>Data Source=<data_source>;User
Id=<user_id>;Password=<password>;LSProvider=ODP;</
CONNECTSTRING>
```

For Microsoft SQL databases:

```
<CONNECTSTRING>Data Source=<address>;Initial
Catalog=<SQL_database>;User
Id=<user_id>;Password=<password>;MultipleActiveResultSets=True;
LSProvider=MSSQL;</CONNECTSTRING>
```

where:

- <data_source> is the Oracle TNS Name for the data source, from the TNS_NAMES.ora file (typically located in the \\<machine>\oracle\network\admin directory)
- <user_id> is the user ID for the database connection.

Note: The user ID specified must have full access to all database objects, including tables, stored procedures, and database triggers. Using a user ID that is either identical to the Qualifier or that is assigned to the PWRLINE_USER role will ensure this.

- <password> is the password for the supplied user ID.
 - <address> is the IP address or Hostname of the MS SQL Server database server
 - <SQL_database> is the name of the MS SQL Server database
3. *Optional.* Select **Data Access** from the Data Sources navigation pane, and configure data access for the data source.
 4. *Optional.* Select **Features** from the Data Sources navigation pane, and configure Data Navigator features (tables and columns) for the data source.
 5. Click **Save**.

Groups

Next, set up groups and assign appropriate Data Access and Feature privileges to each using the Groups tab of the Security user interface. Perform the following steps for each group you need to set up:

1. Click **Add Group** on the Groups tab.

2. Enter a **Name** for the Group and click **Save**.
3. *Optional.* Select **Data Access** from the Groups navigation pane, and grant appropriate data access privileges to the group.
4. Select **Features** from the Groups navigation pane, and select permissions for the group in the Features Privileges Tree. If a plus sign ("+") appears next to an option, click it to expand and display sub-options.

Note: The specific permissions available are dependent on the specific products available.

5. Click **Save**.

Users

Lastly, set up individual users using the Users tab of the Security user interface. Perform the following steps for each user you need to set up:

1. Click **Add User** on the Users tab.
2. Enter a **User ID**, **Password** (in the **Password** and **Confirm Password** fields), and an optional profile for the user and click **Save**.
3. *Optional.* Select **Profiles** from the Users navigation pane, and create additional profiles for the user.
4. *Optional.* Select **Data Access** from the Users navigation pane, and grant appropriate data access privileges to the user.
5. Select **Features** from the Users navigation pane, and select permissions for the user in the Features Privileges tree. If a plus sign ("+") appears next to an option, click it to expand and display sub-options.

Note: The specific permissions available are dependent on the specific products available.

6. Click **Save**.

Setting Up Security in LDAP Mode

Setting up Security in LDAP mode involves the following steps:

1. **Set Up Security Configuration File**
2. **Encrypt LDAP Password**
3. **Add Security Administrator to Security Database**
4. **Administrator Login**
5. **Set Up Security Data**

Set Up Security Configuration File

The Security functionality is enabled by the presence of the LSSECURE.CFG.XML configuration file installed in the **C:\LODESTAR\CFG** directory.

When running security in LDAP mode, the LSSECURE.CFG file contains three elements: the data source used by the Security functionality (required), an optional session expire time (in minutes), and specifics concerning the LDAP server(s) (required).

The data source specified is the data source that contains the security data (Data Sources, Groups, and Users). The LDAP server is the directory access server that stores user authentication information (user IDs and passwords). If not included, the expire time defaults to 60 minutes.

LSSECURE.CFG Example - LDAP Mode

```
<LSSECURE>
  <DATASOURCE>
    <NAME>DatabaseName</NAME>
    <CONNECTSTRING>DSN=DSN;UID=UserName;PWD=Password;</CONNECTSTRING>
    <QUALIFIER>DatabaseQualifier</QUALIFIER>
  </DATASOURCE>
  <EXPIRETIME>60</EXPIRETIME>
  <LDAP>
    <SERVER>
      <HOST>dc1.ldap.lan</HOST>
      <VERSION>3</VERSION>
      <PORT>389</PORT>
      <TIMEOUT>30000</TIMEOUT>
      <UDN>CN=Administrator,CN=Users,DC=ldap,DC=lan</UDN>
      <PWD>...</PWD>
      <USERSEARCH>
        <BASEDN>CN=Users,DC=ldap,DC=lan</BASEDN>
        <SCOPE>ONE</SCOPE>
        <UIDATTRNAME>sAMAccountName</UIDATTRNAME>
        <FILTER>(objectClass=user)</FILTER>
      </USERSEARCH>
    </SERVER>
  </LDAP>
</LSSECURE>
```


Important

This file **MUST** be edited to specify the security data source prior to running Oracle Utilities applications. See **LSSECURE.CFG.XML** on page 2-42 for more information about setting up this file.

Encrypt LDAP Password

Before logging on to the Oracle Utilities application, the LDAP password (contained in the <PWD> element in the LSSECURE.CFG file) must be encrypted using the LSSecureInit.exe command line program. This program must be run on the web/application server where the Oracle Utilities applications are installed. The syntax for this program is as follows:

```
LSSecureInit -s <host> -u <userdn> -p <password>
```

where:

Parameter	Description
-s	<host> is the Fully Qualified Domain Name (FQDN) or IP address of the LDAP server. This should be the same as the <HOST> element in the LSSECURE.CFG file.
-u	<userdn> is the user DN from the <UDN> element in the LSSECURE.CFG file.
-p	<password> is the user password to be encrypted and stored in the <PWD> element in the LSSECURE.CFG file.

Note: This step can be performed at the same time as when adding the Security administrator to the Security Database.

Add Security Administrator to Security Database

Before logging on to the Oracle Utilities application, the LDAP user ID of the security administrator (the person responsible for setting up the Security data) must be added to the Oracle Utilities Data Repository using the LSSecureInit.exe command line program. This program assigns privileges to the specified userid for a Data Source named **SECURITY INITIAL** which grants access to all the Security functions. This program must be run on the web/application server where the Oracle Utilities applications are installed. The syntax for this program is as follows:

```
LSSecureInit -i <userid>
```

where:

Parameter	Description
-i	<userid> is the required LDAP user ID for the Security administrator (the person responsible for setting up the Security data). This should be a valid LDAP userid.

Note: This step can be performed at the same time as when encrypting the LDAP password in the LSSECURE.CFG file.

Administrator Login

When logging into the Security user interface for the first time (before setting up any users, groups, etc.), administrators should login using the security administrator user ID specified when adding the Security administrator the Security Database (above).

This userid is assigned is assigned feature privileges for a Data Source named **SECURITY INTITAL**, and has access to all the Security functions. This enables a first-time user of the

Security user interface to set up the required data sources, groups, and users. Oracle recommends that you set up an alternate security administrator user that has access to all the Security functions for general use.

Important Note

Do not delete the “admin” user or the SECURITY INITIAL data source until and/or unless you have created an alternate security administrator user that has access to all the Security functions.

Set Up Security Data

Setting up Security Data is done in much the same way in LDAP mode as it is in default mode. The only difference is in adding users when in LDAP mode. See **How to add a new user in LDAP mode:** on page 7-38 in the *Oracle Utilities Energy Information Platform User's Guide* for more information.

Integrating Security with LDAP

This section provides additional information helpful when integrating Oracle Utilities Energy Information Platform Security with Lightweight Directory Access Protocol (LDAP), including:

- **Overview**
- **Integration Details**
- **Troubleshooting**
- **Useful Tools**
- **Connecting to LDAP Servers over SSL**

Overview

Lightweight Directory Access Protocol (LDAP) is a standard means of accessing a central directory service used to store user authentication information (and other directory resource information). When security is implemented in LDAP mode, user authentication data is stored in an external LDAP server instead of the Oracle Utilities Data Repository. All other security-related data, including groups, data sources, and sessions are still stored in the Security schema when in LDAP mode.

Integration Details

The integration of security and LDAP consists of:

- Sharing user IDs between the LDAP directory service and the security database (during administrative setup), and
- Performing user authentication on the LDAP directory service during login

Those activities depend upon the following low-level steps:

- Establishing a connection with the directory service
- Searching the necessary entries in the directory tree

Establishing a Connection to the LDAP Directory Service

The TCP connection with the LDAP server(s) should be established prior to taking any directory service related actions (such as creating User IDs). This step uses values from the HOST and PORT elements of the LSSECURE.CFG.XML configuration file. When an attempt to establish a connection fails, the security framework will use next LDAP server within the same group until all servers are utilized.

Any directory service requires binding to it under certain credentials to enable access to any data from the service. Therefore the security framework performs this binding after the TCP connection is established when the UDN and PWD elements are provided in LSSECURE.CFG.XML configuration file, using the values of the UDN and PWD elements as credentials for binding. Due to the necessity to work with different LDAP vendors, a simple authentication method has been chosen that is widely supported by all LDAP directory services. This method leads to following requirements:

- The LDAP server(s) should support simple authentication
- The UDN element should contain the DN for the entry of the binding user. For example, this should be "CN=Administrator,CN=Users,DC=ldap,DC=lan".

It is insufficient to provide just user ID "Administrator" value.

Search Entries

Before any processing can be performed, the appropriate entry in the directory tree must be located. The entry search occurs according to the content of the `USERSEARCH` element in the `LSSECURE.CFG.XML` configuration file, in the following order.

- The search starts from one entry (base-entry) with the DN (distinguished name) equal to the `BASEDN` element in the `LSSECURE.CFG.XML` configuration file.
- The search progressed down through the LDAP tree in one of three possible scopes (according to the values of the `SCOPE` element in the `LSSECURE.CFG.XML` configuration file.)
- The search uses the filter statement created in runtime according to the content of the `UIDATTRNAME` and `FILTER` elements in the `LSSECURE.CFG.XML` configuration file. This filter is created according to the following concatenation rule.

```
ResultFilter = "(&(UIDATTRNAME_value=SearchValue) (FILTER_value))"
```

For example to work with Microsoft Active Directory the configuration file could contain:

```
<UIDATTRNAME>SAMAccountName</UIDATTRNAME>
<FILTER>& (objectClass=user) (memeberOf=CN=Lodestar
Support,CN=Users,DC=ldap,DC=lan)</FILTER>
```

The resulting filter statement will be:

```
ResultFilter =
"(&(sAMAccountName=Search_Value) (&(objectClass=user)
(memeberOf=CN=Lodestar Support,CN=Users,DC=ldap,DC=lan)))".
```

Note: the `Search_Value` depends on the purpose of the search. When loading users from the LDAP directory service to the security database, this is equal to `"*"`. During the user authentication this is equal to the `"User ID"` passed from the login screen.

Sharing User IDs

Even after integration with the LDAP directory service, the security database provides the possibility to store related user characteristics such as user properties, permissions, grouping information, and so forth. User IDs should be populated from the directory service enable association with such data, and occurs in the following order:

1. Establishing a connection to LDAP server(s) as described in **Establishing a Connection to the LDAP Directory Service** on page 12-23.
2. Performing a search of user entries in the LDAP directory tree as described in **Search Entries** on page 12-24.
3. User creation in the security database for the chosen directory entry. The UID of the new user will be equal to the value of the entry's attribute specified in the UIDATTRIBUTE in the LSSECURE.CFG.XML configuration file. Oracle Utilities Energy Information Platform Security is not responsible for passwords when using LDAP for user authentication.

User Authentication

User authentication involves an attempt to bind to the LDAP server under the user's credentials. As described in **Establishing a Connection to the LDAP Directory Service** on page 12-23, the DN of the user's entry in the LDAP directory is required for binding. Therefore the authentication process consists of following steps.

1. User's entry search as it is described in **Search Entries** on page 12-24. The Search_Value in that case is a User ID provided by end-user on the login screen. User authentication continues only if this step is successful.
2. Binding to LDAP server. The DN of the entry obtained in the previous step and password passed from login screen are used as user's credentials.

Authentication is considered successful when binding step is passed successfully.

Troubleshooting

Different configuration issues could impact the implementation and administration of security when integrating with LDAP. The LSSecureInit application can be used to check the entire configuration of the security framework. This program must be run on the web/application server where the Oracle Utilities applications are installed. The syntax for this program is as follows:

```
LSSecureInit -t [-tu <userid> -tp <password> -ts <useridmask> [-tn <count>]]
```

where:

Parameter	Description
-t	Test mode flag. Indicates that the application is being used for testing.
-tu	<userid> is the user ID for testing the LDAP server login. This should be a valid LDAP userid.
-tp	<password> is the user password for the supplied user ID (<userid>).
-ts	<useridmask> is the mask for the user id to perform user searches
-tn	<count> is the number of tests to perform. The default is 1.

When run in test mode, LSSecureInit tests the following:

1. Database connection to the security schema
2. Connection to LDAP servers
3. User authentication for the provided credentials
4. User search for the provided user id mask

Steps 3 and 4 can be repeated the requested number of times (based on the value of the -tn parameter).

Useful Tools

Knowledge of the LDAP directory hierarchy is very useful during all steps of implementation and administration when integrating security with LDAP. There are number of tools that can help to check the connection to the directory service and to explore the directory tree.

- **LDPEXE:** This is a Windows 2000 Support Tools utility used to perform LDAP searches. See <http://support.microsoft.com/kb/224543/EN-US/>
- **adsiedit.msc:** This GUI tool is a Microsoft Management Console (MMC) snap-in that acts as a low-level editor for Microsoft Active Directory. See <http://www.microsoft.com/resources/documentation/WindowsServ/2003/all/techref/en-us/Default.asp?url=/Resources/Documentation/windowsserv/2003/all/techref/en-us/adsiedit.asp>

Connecting to LDAP Servers over SSL

Oracle Utilities Energy Information Platform Security can be integrated with LDAP-compliant directory servers using Secure Sockets Layer (SSL) technology.

Required Certificates

- When using Windows 2000, only the LDAP server(s) requires a server certificate.
- When using Windows 2003 (or Windows XP), the Certification Authority issued server certificate must be registered in the Trusted Root Certification Authorities list on the web server.

Enabling the SSL Connection

To enable connection to an LDAP server via SSL, set the <PORT> and <HOST> elements in the LSSECURE.CFG.XML file as follows:

PORT: Set this to the port used to support secure connections over SSL on the server. This is usually port 636.

Example:

```
<LSSECURE>
  <DATASOURCE>
    ...
  </DATASOURCE>
  <LDAP>
    <SERVER>
      ...
      <PORT>636</PORT>
      ...
    </SERVER>
  </LDAP>
</LSSECURE>
```

HOST: Set this to the value of the "Issued to" field on the General tab of the Certificate dialog for the server certificate on the LDAP server.

To view the server certificate on the client machine (the web server in this case) use the following procedure:

1. Open a new instance of Internet Explorer.
2. Enter the address for the LDAP server as follows:
https://<LDAP_SERVER_IP_ADDRESS>:<SSL_PORT>
where:
 - <LDAP_SERVER_IP_ADDRESS> is the IP address of the LDAP server.
 - <SSL_PORT> is the SSL port number (typically 636).
3. If the "Choose a digital certificate" window opens, click Cancel.
4. If the "Security Alert" window opens, click View Certificate.
5. The Certificate dialog opens, displaying the server certificate.
6. Enter the value in the "Issued to" field in the <HOST> element in the LSSECURE.CFG.XML file.

Example:

```
<LSSECURE>
...
<LDAP>
  <SERVER>
    <HOST>dc1.ldap.lan</HOST>
  ...
</SERVER>
</LDAP>
</LSSECURE>
```

Client certificate required on LDAP Server

If the LDAP server requires a client certificate to connect via SSL, the client certificate must be installed in the Personal Store on the web server.

In addition, the `<CERT*THUMBPRINT>` element in the LSSECURE.CFG.XML file must be specified as follows:

CERT*THUMBPRINT: Set this to the value of the Thumbprint property (a hexadecimal string) on the Details tab of the Certificate dialog for the server certificate.

Example:

```
<LSSECURE>
  <DATASOURCE>
    ...
  </DATASOURCE>
  <LDAP>
    <SERVER>
      ...
      <CERTTHUMBPRINT> 5a e9 22 4a 07 04 ea 4b 9c 82 d5 4b 80 a1 3f 84 ed 46 62
63 </CERTTHUMBPRINT>
      ...
    </SERVER>
  </LDAP>
</LSSECURE>
```


Single Sign-On

The Oracle Utilities Energy Information Platform supports single sign-on (SSO) capabilities, allowing for user authentication against a central directory server and thereby bypassing the login screen normally used to access the Energy Information Platform.

This section describes the single sign-on capabilities, including:

- **LSSECURE.CFG.XML File**
- **Creating Energy Information Platform User IDs for Single Sign-On**
- **IIS Configuration**

LSSECURE.CFG.XML File

The LSSECURE.CFG.XML configuration file supports an element to define the means in which Microsoft Internet Information Service (IIS) should pass a user's User ID to the Energy Information Platform application.

Sample LSSECURE.CFG.XML File:

```
<LSSECURE>
  <DATASOURCE>
    ...
  </DATASOURCE>
  <SSO FAILOVER="IGNORESSO">LOGON_USER</SSO>
  ...
</LSSECURE>
```

Element Description:

SSO: Defines a parameter used by Microsoft Internet Information Service (IIS) to pass the user's User ID to the security component for purposes of user authentication.

- To use Windows Integrated Authentication, specify "**LOGON_USER**" in the <SSO> element.

Note: IIS must be configured to support Windows Integrated Authentication. See **IIS Configuration** on page 12-30 for more information.

- To use an HTTP Header request parameter (used with SSO systems, such as SiteMinder), specify "**HTTP_<HEADER>**" in the <SSO> element, where <HEADER> is the name of the header request parameter used by the SSO system. For example, if using SiteMinder, you would specify "**HTTP_SMUSER**" in the <SSO> element.

Note: The <HEADER> parameter must NOT contain dashes (-) or underscore characters.

Attributes:

- **FAILOVER:** Specifies the behavior in the event of failure when connecting to the user authentication directory server. To default to the current non-single sign-on user authentication method (LDAP mode or default mode, set this attribute equal to "IGNORESSO" (FAILOVER="IGNORESSO"). To fail authentication if SSO is missing or empty, omit this attribute or set this attribute equal to NULL (FAILOVER="").

Creating Energy Information Platform User IDs for Single Sign-On

When creating Energy Information Platform user IDs for use with single sign-on, use the following guidelines:

- For Windows Integrated Authentication:
 - **User ID:** Must be a <domain>\<user_id> pair, as it appears in the Windows Security configuration screens.

For example, to specify the “LSUSER” user id for the “Galaxy” domain, you would create the following Energy Information Platform user ID: “Galaxy\LSUSER”
 - **Password:** A non-empty string (the password defined for the Energy Information Platform user ID is ignored; the user’s Windows password is used for authentication).
- For HTTP Header request parameter:
 - **User ID:** Must be a user ID as registered in the SSO system.

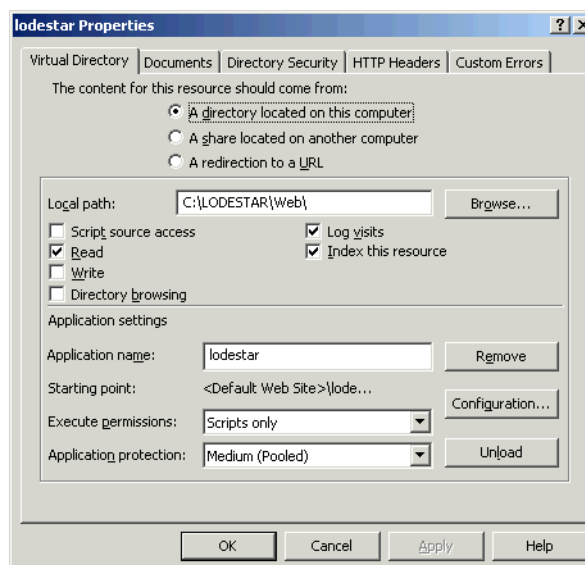
For example, to specify the “LSUSER” user id defined in an SSO system, you would create the following Energy Information Platform user ID: “LSUSER”
 - **Password:** A non-empty string (the password defined for the Energy Information Platform user ID is ignored; the user’s Windows password is used for authentication).

IIS Configuration

When using Windows Integrated Authentication, you must also configure IIS by disabling Anonymous Access and enabling one of Authenticated Access options.

How to configure IIS for Windows Integrated Authentication Single Sing-On:

1. Select **Start→Administrative Tools→Computer Management**.
2. Expand **Services and Applications**.
3. Expand **Internet Information Services (IIS Manager)**.
4. Expand **Websites**.
5. Expand **Default Website**.
6. Select “Iodestar,” click the right mouse button and select **Properties**. The **Iodestar Properties** dialog opens.



7. Select the **Directory Security** tab on the **lodestar Properties** dialog.
8. Click **Edit** in the **Anonymous and Authenticated Control** box. The **Authentication Methods** dialog opens.
9. Uncheck the **Anonymous Access** checkbox.
10. Select one of the checkboxes under **Authenticated Access**.
11. Click **OK** to close the **Authentication Methods** dialog.
12. Click **OK** to close the **lodestar Properties** dialog.

Chapter 13

Energy Information Platform Reporting Framework

This chapter describes the features of the Oracle Utilities Energy Information Platform Reporting Framework, including:

- An **Overview** of the Energy Information Platform Reporting Framework
- **Energy Information Platform Reporting Framework Database Tables**
- **Running and Viewing Reports**
 - Running Reports from the User Interface
 - Running Reports from the Oracle Utilities Rules Language
 - Running Reports in Batch Mode
 - Viewing Reports
- **Configuring the Energy Information Platform Reporting Framework for use with Oracle BI Publisher**
- **Adding Oracle BI Publisher Reports to the Energy Information Platform**
- **Designing Oracle BI Publisher Reports for use with the Reporting Framework**
- **Adding Custom Crystal Reports to the Oracle Utilities Energy Information Platform**
- **Designing Crystal Reports for use with the Reporting Framework**
- **Adding Oracle Utilities Rules Language Rate Schedules to the Energy Information Platform**
- **Reporting on Interval Data**

Overview

The Oracle Utilities Energy Information Platform Reporting Framework provides a framework for generating and viewing reports in the Oracle Utilities Energy Information Platform. This includes running and viewing Oracle Utilities-supplied reports (such as those provided with Oracle Utilities products such as Oracle Utilities Billing Component or Oracle Utilities Quotations Management), as well as running and viewing custom reports created using Oracle BI Publisher or Crystal Reports.

This section includes the following:

- **Reporting Framework Components**
- **How the Energy Information Platform Reporting Framework Works**
- **Supported Report Formats**

Reporting Framework Components

The Energy Information Platform Reporting Framework uses the following components:

Database Tables

These are tables in the Oracle Utilities Data Repository that store data used by the Reporting Framework. See **Energy Information Platform Reporting Framework Database Tables** on page 13-6 for more information.

Report Manager

The Report Manager is the engine that runs the Reporting Framework, including creating input parameter screens (for custom reports), creating report requests, and updating the State of reports being processed. See **How the Energy Information Platform Reporting Framework Works** on page 13-4 for more information about the report manager's role in the Reporting Framework.

Report Monitor Service

The Report Monitor Service is a Windows service that monitors the Report Instance table in the Oracle Utilities Data Repository (see **Energy Information Platform Reporting Framework Database Tables** on page 13-6) for reports to be run. See **How the Energy Information Platform Reporting Framework Works** on page 13-4 for more information about the report monitor service's role in the Reporting Framework.

The specific data source(s) monitored by the Report Monitor Service are specified in the LSREPORTMONITOR.CFG.XML file. See **LSREPORTMONITOR.CFG.XML** on page 2-35 for more information about this file.

Report Generators

Report generators are executable applications that generate reports. There are three supported report generators:

- **Report Generator (LSReport.exe):** Used to generate Oracle Utilities Billing Component and Rules Language reports.
- **Oracle BI Publisher (LSBIReport.exe):** Used to generate reports created using Oracle BI Publisher, including user-created custom reports.
- **Crystal Reports:** Used to generate reports created using Crystal Reports, including user-created custom reports.
- **Active Reports:** Used to generate reports created using Active Reports (Oracle Utilities Receivables Component reports only).

- **Load Analysis** (CsRptGenerator.exe): Used to execute Oracle Utilities Load Analysis programs.

Report Templates

Report templates define the queries and parameters used to generate reports, including Oracle BI Publisher reports, Crystal Reports, Active Reports, and Oracle Utilities Rules Language

User Interface

The user interface is the web-enabled Oracle Utilities Energy Information Platform that allows users to administer, generate, and view reports, and includes the following types of screens:

- **Report Administration screens:** Screens used to search, view, edit, and add report templates, report relationships, and report categories. See **Report Administration** on page 7-55 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about using the Report Administration screens.
- **Run Reports screens:** Screens accessible from individual Oracle Utilities applications that list the reports available through the Reporting Framework for that particular product. See **Run Reports** on page 7-74 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about running reports.
- **Report Parameter screens:** Screens that allow users to enter parameters for running reports.
- **View Reports screens:** Screens that list reports related to a specific product (Oracle Utilities Billing Component, Oracle Utilities Quotations Management, etc.). See **View Reports** on page 7-77 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about viewing reports.
- **Report Viewers:** Report-type specific viewers used to display reports.

How the Energy Information Platform Reporting Framework Works

There are two primary functions that users perform with the Reporting Framework. These include generating reports, and viewing reports.

Generating Reports

The following steps outline the process the Report Framework follows to generate reports.

1. The user selects the report to run from either the Left menu or a Run Reports screen.
2. The Report Manager displays or creates an Active Server Page (ASP) page in which users can enter report parameters.
3. The user enters the report parameters and triggers the report, sending a report request to the Report Manager.
4. The Report Manager creates a report request record in the Report Instance table, including the following data:
 - Report Type (PLBX (Oracle Utilities Billing Component), LSRate (Oracle Utilities Rules Language), LSIMPORT, Crystal, BIPublisher, Active, or LoadAnalysis)
 - Report Category
 - Report Parameters
 - State (when first created, all report request records have a State of QUEUED).
5. The Report Monitor Service detects the report request in the Report Instance table, changes the report's State to INIT (Initialized), and triggers the appropriate Report Generator for the report (Oracle Utilities Billing Component or Rules Language, Crystal, BIPublisher, Active, or LoadAnalysis).
6. The Report Manager changes the report's State to PROCESS and the Report Generator generates the report. If errors occur during processing, the Report Manager updates the report's State to ERROR.
7. When the report generation is complete, the report data is stored in the Report Unit table and the Report Manager updates the State to DONE.

Viewing Reports

The following steps outline how users view reports using the Report Framework.

1. A user selects the View Reports option on the web interface. Each product has its own View Reports option (Oracle Utilities Billing Component, Oracle Utilities Receivables Component, Oracle Utilities Quotations Management, etc.). Reports are associated with a specific product in the Report Template Product table.
2. The View Reports window opens displaying the user's reports. A user's reports include all reports generated by the user and all shared reports.
3. To view a specific report, the user selects the report (by clicking on the Report Title link). The selected report opens in a type-specific report viewer.
4. The user can delete reports from the View Reports window. When a user deletes a report, the Report Manager changes the report's State to DELETING. This deletes (removes) the report record from the database, and the report no longer appears on the View Reports window.

Supported Report Formats

The Reporting Framework supports the following types of reports:

- **Oracle Utilities Reports:** Reports created by Oracle Utilities products, including Oracle Utilities Billing Component and the Oracle Utilities Rules Language.
- **Oracle BI Publisher:** Reports created using Oracle BI Publisher, including both Oracle Utilities-supplied reports and user-created custom reports.
- **Crystal Reports:** Reports created using Business Objects' Crystal Reports, including both Oracle Utilities-supplied reports and user-created custom reports.
- **Active Reports:** Reports created using Active Reports (Oracle Utilities Receivables Component reports only).
- **Load Analysis:** Reports created by Oracle Utilities Load Analysis programs.

Report Output Formats (Oracle BI Publisher and Crystal Reports Only)

Oracle BI Publisher reports and Crystal Reports can be produced in a number of output formats, including Adobe Acrobat (PDF), Microsoft Word (*.doc), Microsoft Excel (*.xls), and Rich Text Format (*.rtf). The table below lists the available output formats. By default a number of formats are disabled. The default behavior can be overridden by editing the LSREPORTMONITOR.CFG.XML file.

Format	Default
Adobe Format	Enabled
Microsoft Excel	Enabled
Microsoft Excel (Data only)	Disabled
Microsoft Excel 2000	Disabled
Microsoft PowerPoint	Disabled
Microsoft Word	Enabled
Rich Text Format	Enabled
Separated Values	Enabled
Web Archive (MHTML)	Enabled
XML	Enabled

Energy Information Platform Reporting Framework Database Tables

The Energy Information Platform Reporting Framework uses several tables in the Oracle Utilities Data Repository. This section describes these tables and the data stored in each.

A Note about Predefined Records:

Several of the tables described in this section contain predefined records used by the Report Framework. These records are listed in the appropriate table description. **Do not modify or delete any of these predefined records.**

Report Category

Records in the Report Category table store user-defined report categories. Records in this table contain the following information:

- **Report Category:** The name of the category (Daily, Monthly, Billing, Payments, etc.).
- **Description:** A description of the category.

Report Instance

Records in the Report Instance table store individual report records. Records in this table contain the following information:

- **UIDReport:** The unique ID of the report instance.
- **GUID:** The global unique ID of the report instance.
- **Report Name:** The name of the report, defined in the **Report Template** table.
- **Report Type:** The type of report (PLBX (Oracle Utilities Billing Component), LSRate (Oracle Utilities Rules Language), LSIMPORT, Crystal, BIPublisher, Active, LoadAnalysis).
- **Report Title:** The title of the report. For some reports this can be supplied when the report is generated.
- **Report State:** The current state of the report instance. Report instances can have one of the following states:
 - 1 - QUEUED (Queued)
 - 2 - INIT (Initialized)
 - 3 - PROCESS (In Process)
 - 4 - DONE
 - 5 - ERROR
 - 6 - DELETING
- **Report Progress:** The current percentage of the report instance that has been generated, from 0 to 100 percent.
- **Process ID:** The server and process ID of the process that is generating the report instance.
- **Input Parameters:** Input parameters for the report.
- **Is Shared:** A flag that indicates if the report instance is shared by all users or is accessible only the user who generated the report.
- **User ID:** The User ID of the user who created the report instance.
- **Creation Date:** The date and time on which the report instance was created.
- **Is Base:** A flag that indicates if the report is supplied by Oracle Utilities or is a custom report.

- **Output Format:** The output format for the report instance. Used only with Crystal Reports. Can be one of the following:
 - Adobe Acrobat
 - Microsoft Excel
 - Microsoft Excel (data only)
 - Microsoft Word
 - Rich Text Format
 - Separated Values (CSV)
 - XML
- **Priority:** The priority of the report instance, from the **Report Priority** table.

Report Instance Log

Records in the Report Instance Log table capture changes in the status of individual report records in the **Report Instance** table. For example, when a report changes state from QUEUED to INIT, a record is inserted in this table to log the change in the report instance. When the report state changes from INIT to PROCESS, another record is inserted to log the change.

Records in this table are created by the LSRFRPTINSTANCE_LOG_TRG database trigger. This trigger is disabled by default. This trigger can be enabled by database administrators via database access tools.

Note: This trigger should ONLY be enabled when troubleshooting issues with the Reporting Framework, and should be disabled in production environments.

Note: This table is not accessible from Data Navigator or Data Manager. To view records in this table, use database access tools.

Records in this table contain the following information:

- **UIDLog:** The unique ID of the log record.
- **Timestamp Log:** The date and time of when the log record was created. This indicates the date and time when the state of the parent report instance record was changed.
- **UIDReport:** The unique ID of the parent report instance.
- **GUID:** The global unique ID of the parent report instance.
- **Report Name:** The name of the report, defined in the **Report Template** table.
- **Report Type:** The type of report (PLBX (Oracle Utilities Billing Component), LSRate (Oracle Utilities Rules Language), LSIMPORT, Crystal, BIPublisher, Active, LoadAnalysis).
- **Report Title:** The title of the report. For some reports this can be supplied when the report is generated.
- **Report State:** The current state of the parent report instance. Report instances can have one of the following states:
 - 1 - QUEUED (Queued)
 - 2 - INIT (Initialized)
 - 3 - PROCESS (In Process)
 - 4 - DONE
 - 5 - ERROR
 - 6 - DELETING

- **Report Progress:** The current percentage of the parent report instance that has been generated, from 0 to 100 percent.
- **Process ID:** The server and process ID of the process that is generating the parent report instance.
- **Is Shared:** A flag that indicates if the parent report instance is shared by all users or is accessible only the user who generated the report.
- **Is Base:** A flag that indicates if the report is supplied by Oracle Utilities or is a custom report.
- **User ID:** The User ID of the user who created the parent report instance.
- **Creation Date:** The date and time on which the parent report instance was created.
- **Output Format:** The output format for the parent report instance. Used only with Crystal Reports. Can be one of the following:
 - Adobe Acrobat
 - Microsoft Excel
 - Microsoft Excel (data only)
 - Microsoft Word
 - Rich Text Format
 - Separated Values (CSV)
 - XML
- **Priority:** The priority of the parent report instance, from the **Report Priority** table.

Report Priority

Records in the Report Priority table define priorities for processing reports via the Reporting Framework. Report requests are processed in order of priority (High, Medium, Low). Records in this table contain the following information:

- **Priority:** A number that represents the priority. Smaller numbers are considered to have a higher priority than larger numbers.
- **Description:** A description of the priority.

The Report Priority table includes the following predefined records:

Priority	Description
1	High
2	Medium
3	Low

Report Template

Records in the Report Template table store information about specific report templates. Templates are Crystal Report templates used by the Reporting Framework. Records in this table contain the following information:

- **Report Name:** The name of the report.
- **Report Type:** The type of report (BIPublisher, Crystal, or LSRate).
- **Description:** A description of the report template.

- **Generate ASP:** A flag that indicates if the Report Framework should automatically generate an input parameters page when the user runs the report.
- **Is Base:** A flag that indicates if the report is supplied by Oracle Utilities or is a custom report.
- **Allow Synchronization:** A flag that indicates if reports based on the report template can be run in real-time or if they are to be queued.
- **User ID:** The User ID of the user who created the report template, or who last updated the report template.
- **Creation Date:** The date and time on which the report instance was created.
- **Report Template:** The path and file name of the report template file.
- **ASP Template:** The path and file name of a hard-coded ASP page associated with the report template.
- **Output Format:** The output format for the report. Used only with Crystal Reports. Can be one of the following:
 - BLANK (indicates that the output format is specified when the report is viewed).
 - Adobe Acrobat
 - Microsoft Excel
 - Microsoft Excel (data only)
 - Microsoft Word
 - Rich Text Format
 - Separated Values (CSV)
 - XML
- **Save To:** Specifies where the report is saved to upon execution. Can include a path and file name.

Report Template Product

Records in the Report Template Product table store relationships between report templates, report categories, and Oracle Utilities products. Records in this table contain the following information:

- **Report Name:** The name of the report, defined in the **Report Template** table, or as pre-defined by Oracle Utilities.
- **Product Code:** The product the report template is associated with.
- **Report Category:** The name of the report category (Daily, Monthly, Billing, Payments, etc.).

The Report Template Product table includes the following predefined records:

Report Name	Product Code
ConvertIntervalData	TU
SecurityPermissionsbyGroup	TU
SecurityPermissionsbyUser	TU
WQAging	WQ
WQOpenByUser	WQ
WQProductivty	WQ

Report Name	Product Code
IMPORT	TU

When specific products are installed, additional records are inserted into the Report Template Product table.

Report Type

Records in the Report Type table store report types. Records in this table contain the following information:

- **Report Type:** The report type (LSRate, Crystal, BIPublisher, Active).
- **Description:** A description of the report type.

The Report Type table includes the following predefined records:

Report Type	Description
PLBX	Oracle Utilities Rules Language Reports
LSRate	Oracle Utilities Rules Language
LSRunLS	Run Rules Schedule
LSIMPORT	Import
Crystal	Crystal Reports
BIPublisher	Oracle BI Publisher
Active	Active Reports
LOAD_ANALYSIS	LoadAnalysis

Report Unit

Records in the Report Unit table store generated reports. Some reports (such as those generated by Oracle Utilities Billing Component) can be generated in multiple sub-sections (each associated with a business unit such as an Account or Customer), called units. Crystal and Active reports have only one unit. Records in this table contain the following information:

- **UIDREPORT:** The unique ID of the report instance associated with the report unit (from the **Report Instance** table).
- **Unit Index:** The index of the report unit.
- **Content:** The content of the report unit.

The Report Administration User Interface

The Report Administration user interface is accessed via the **Report Admin** option under the **Tools and Utilities** menu. Selecting that option opens the Report Administration screen.

See **Report Administration** on page 7-55 in the *Oracle Utilities Energy Information Platform User's Guide* for details on using the Report Administration user interface.

Running and Viewing Reports

This section describes how users run and schedule, and view reports in the Oracle Utilities Reporting Framework, including:

- **Running Reports from the User Interface**
- **Running Reports from the Oracle Utilities Rules Language**
- **Running Reports in Batch Mode**
- **Viewing Reports**

Running Reports from the User Interface

See **Run Reports** on page 7-74 in the *Oracle Utilities Energy Information Platform User's Guide* for details on running reports from the Energy Information Platform user interface.

Running Reports from the Oracle Utilities Rules Language

Reports can also be initiated and run from Oracle Utilities Rules Language rate schedules and riders. The CREATEREPORT function allows reports to be initiated during Rules Language processing. See **CREATEREPORT Function** on page 13-108 of the *Oracle Utilities Rules Language Reference Guide* for more information about this function.

Running Reports in Batch Mode

Reports can also be run in batch mode using the RUNREPORT.EXE command line program.

Prerequisites

The Report Template used to generate the report to be run must be defined in the Oracle Utilities Data Repository. See **Templates Tab** on page 7-55 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about creating Report Templates using the Report Administration User Interface.

RUNREPORT Command Syntax

RUNREPORT uses the syntax shown below. Parameter switches are case insensitive (you can enter them in either upper or lower case (-c or -C)).

```
runreport -rt ReportType -rn ReportName -ru ReportUser <DataSource>  
[-t ReportTitle] [<Parameters>] [-i ImmediateGeneration] [-s Shared] [-lcfg logging configuration filename]
```

In actual use, the command must be entered on one line. Also, you must either change to the directory in which the program is stored (typically, \LODESTAR\bin) before entering the command, or specify the path in the command. To view a list of all parameters on-screen, type **runreport -?** at the command prompt.

Parameter	Description
-rt	<i>ReportType</i> is the report type for the report to be run. Can be either BIPublisher (for BIPublisher Reports), Crystal (for Crystal Reports) or LSRate (for Oracle Utilities Rules Language reports). This parameter is required, and is case-sensitive .
-rn	<i>ReportName</i> is the name of the report template exactly as it appears in the Report Name (RPTNAME) column in the Report Template (LSRFRPTTEMPLATE) table. This parameter is required, and is case-sensitive .

Parameter	Description
-ru	<i>ReportUser</i> is the User ID that will appear in the User ID (USERID) column of the Report Instance (LSRFRTPINSTANCE) table. This parameter is required.
<DataSource>	<p>Data source connection for the Oracle Utilities Data Repository. This parameter is required, and can be one of the following:</p> <p>-d <ConnectString> [-q <qual>] where:</p> <ul style="list-style-type: none"> • <ConnectString> is a connect string in one of the following formats: <p>For Oracle databases:</p> <p>"Data Source=<data_source>;User ID=<user_id>;Password=<password>;LSProvider=ODP;"</p> <p>For Microsoft SQL databases:</p> <p>"Data Source=<address>;Initial Catalog=<SQL_database>;User Id=<user_id>;Password=<password>;MultipleActiveResultSets=True;LSProvider=MSSQL;"</p> <p>where:</p> <ul style="list-style-type: none"> • <data_source> is the Oracle TNS Name for the data source, from the TNS_NAMES.ora file (typically located in the \\<machine>\oracle\network\admin directory) • <user_id> is the user ID for the database connection • <password> is the password for the supplied user ID. • <address> is the IP address or Hostname of the MS SQL Server database server • <SQL_database> is the name of the MS SQL Server database • <qual> is an optional database qualifier. The default is PWRLINE. <p>OR</p> <p>-cs <ConnectString> [-q <qual>] where:</p> <ul style="list-style-type: none"> • <ConnectString> is a connect string in one of the formats outlined above • <qual> is an optional database qualifier. The default is PWRLINE. <p>OR</p> <p>-df <XML Data Source file> where:</p> <ul style="list-style-type: none"> • <XML Data Source file> is an XML file containing the data source name, user ID, password, and optional qualifier. If this format is used, you must supply the entire path and file name for the XML file. If a qualifier is used, it must be contained in the XML file.

Parameter	Description
-t	<i>ReportTitle</i> is an optional title for the report. If there are any spaces in the title, it must be entered within quotation marks ("Sample Report Title").
<Parameters>	Optional parameters to be passed to the report. The specific parameters passed are dictated by the report to be run. Any parameters required to run the report are specified here. Can be supplied in one of two forms: <ul style="list-style-type: none"> -pf <path and file name of XML file containing parameters> -ps <XML parameter string> See Working with Report Parameters on page 13-14 for more information about the XML format used for report parameters.
-i	<i>ImmediateGeneration</i> dictates when the report is run. By default the report will be queued. If this parameter is supplied, the report will be run immediately on the machine executing the runreport command.
-s	<i>Shared</i> dictates if the report is shared or not. By default reports are not shared, and are only available to the user whose User ID is supplied in the -ru parameter. If this parameter is supplied, the report is shared.
-lcfg	<i>logging configuration filename</i> Name of an optional logging configuration file that specifies where error and log messages are sent. If you omit this parameter, the application creates a log file named RUNREPORT.LOG in the LOG directory.

Working with Report Parameters

Parameters are passed to the RUNREPORT.EXE as either an XML file or an XML string. In both cases, the XML format is the same. The structure of the XML is based on the type of report (Crystal or Oracle BI Publisher)

Crystal Reports Report Parameters

```
<CRParameters>
  <CRParameter ParameterFieldName= "Tail name" CurrentValue= "Tail value"/>
</CRParameters>
```

Oracle BI Publisher Report Parameters

```
<BIPParameters>
  <BIPParameter ParameterFieldName="Tail name">
    <CurrentValue>Tail value</CurrentValue>
  </BIPParameter>
```

where:

- "Tail Name" is the tail name (in quotes)
- "Tail Value" is the tail value (in quotes)

If the parameter to be passed is a table value, the ParameterFieldName must be identified using the TABLE.COLUMN format,

where

- TABLE is the name of the table
- COLUMN is the name of the column.

For example, the Work Queue Type Code (WQTYPECODE) column in the Work Queue Type (LSWQTYPE) table would be passed as "LSWQTYPE.WQTYPECODE".

Example

To pass the following parameters to a report:

- Work Queue Type: ABORT
- Work Queue: ACCOUNTNOTE
- Assigned to User ID: lou_p
- Product Code: BE (Oracle Utilities Billing Component)
- Start Time: 12/04/2004 00:00:00
- Stop Time 12/04/2004 223:59:59

you would create the following XML structure as either an XML file or an XML string:

```
<CRParameters>
  <CRParameter ParameterFieldName="LSWQTYPE.WQTYPECODE" CurrentValue="ABORT"/>
  <CRParameter ParameterFieldName="LSWQUEUE.WQQUEUECODE"
CurrentValue="ACCOUNTNOTE"/>
  <CRParameter ParameterFieldName="ASSIGNEDTOUSERID" CurrentValue="lou_p"/>
  <CRParameter ParameterFieldName="LSPRODUCT.PRODUCTCODE" CurrentValue="BE"/>
  <CRParameter ParameterFieldName="STARTTIME" CurrentValue="2004-12-04
00:00:00"/>
  <CRParameter ParameterFieldName="STOPTIME" CurrentValue="2004-12-04
23:59:59"/>
</CRParameters>
```

Example

The following example would run the Work Queue Aging report (WQAging) on a data source defined in the LSDataSource.XML file. The report run uses parameters defined the WQ_Aging_Params.xml file, and will be shared by all users.

```
RUNREPORT -df C:\LODESTAR\DS\LSDataSource.xml -rt Crystal -rn WQAging
-ru lou_p -pf C:\LODESTAR\Params\WQ_Aging_Params.xml -s
```

Viewing Reports

See **View Reports** on page 7-77 in the *Oracle Utilities Energy Information Platform User's Guide* for details on viewing reports from the Energy Information Platform user interface.

Configuring the Energy Information Platform Reporting Framework for use with Oracle BI Publisher

This section describes configuration steps required when using Oracle BI Publisher with the Energy Information Platform Reporting Framework. These steps include configuration tasks on both the Oracle BI Publisher server, as well as the Energy Information Platform.

Note: This version of the Energy Information Platform supports version 10.1.3.4.1 or version 11.1.1 of Oracle BI Publisher.

Oracle BI Publisher Configuration

This section outlines Oracle Business Intelligence Publisher configuration tasks required to enable Oracle Business Intelligence Publisher to generate and display reports run through the Energy Information Platform, including:

- **Start the Oracle BI Publisher Software**
- **Log In to the Oracle BI Publisher Software**
- **Create User for Reporting Framework**
- **Set Report Path**
- **Create Data Source for Oracle Utilities Data Repository**

Note: if installing Oracle BI Publisher on the same machine as the Energy Information Platform application server or web server software, install the Energy Information Platform software before proceeding with the following tasks.

Start the Oracle BI Publisher Software

The first step in configuring the Oracle BI Publisher software is to start the application.

To start the application, select **Start->All Programs->Oracle-BIPHome1->Start BI Publisher**.

A console (DOS) window opens on the desktop. Leave this window open in the background.

Log In to the Oracle BI Publisher Software

Once the Oracle BI Publisher software is running, you must log in to the application to continue the configuration tasks.

1. Launch the BI Publisher application using one of the following methods:

From the Start menu, select **Start->All Programs->Oracle - <ORACLE_HOME_NAME> ->BI Publisher Server**

where:

- **<ORACLE_HOME_NAME>** is the name you gave the Oracle Home where you installed Oracle BI Publisher.

From your browser, open your browser to `http://<hostname>:<port>/xmlpserver`

where:

- **<hostname>** is the name of the server
- **<port>** is the port where you installed Oracle BI Publisher

For example:

`http://localhost:9704/xmlpserver`

2. Log in to the Oracle BI Publisher server using the default administrator user name and password:

- User Name: Administrator
- Password: Administrator

Create User for Reporting Framework

You must create a BI Publisher administrator user on the Oracle BI Publisher server that the Reporting Framework will use to connect to the BI Publisher server when users create, run, and view BI Publisher reports.

To create a user, use the following procedure:

1. Navigate to the Admin tab, and click **Users** (under Security Center).
The User tab of the Security Center screen opens.
2. Click **Create User**.
The Create User screen opens.
3. Enter the User ID and Password for the user and click **Apply**.
The User tab of the Security Center screen now displays the new user.
4. Click the **Assign Roles** icon.
The Assign Roles screen opens.
5. Select **BI Publisher Adminstator** in the Available Roles box, and click **Move**.
The BI Publisher Adminstator role now appears in the Assigned Roles box.
6. Click **Apply**.

Refer to the Oracle BI Publisher documentation for more information about users and roles.

The User ID and Password for this user should be specified in the BI_PUBLISHER/CONNECTSTRING element in the LSReportMonitor.cfg.xml configuration file on each application or web server on which the Reporting Framework will generate Oracle BI Publisher reports.

Set Report Path

You must set the path to the directory where the report template files are stored on the server.

To set the report path, use the following procedure:

1. Identify the report templates folder on the Oracle BI Publisher server.
 - If the Energy Information Platform (and related products) is installed on this machine, this is “C:\lodestar\bip.”
 - If Oracle BI Publisher is installed on a stand-alone server (a server where the Energy Information Platform is NOT installed), create a directory for the report templates, and copy the contents of the “C:\lodestar\bip” directory from an application server or web server to the directory you created.
2. Navigate to the Admin tab, and click **Report Repository** (under System Maintenance).
The Report Repository tab of the System Maintenance screen opens.
3. Select **File System** from the Repository Type drop-down list.
4. Enter the path to the directory where the report templates are located (defined in Step 1 above).

Create Data Source for Oracle Utilities Data Repository

You must also create a JDBC data source on the Oracle BI Publisher to allow a connection to the Oracle Utilities Data Repository. This is used when designing and testing reports.

To create a JDBC data source, use the following procedure:

1. Navigate to the Admin tab, and click **JDBC Connection** (under Data Sources).

The JDBC tab of the Data Sources screen opens.

2. Click **Add Data Source**.

The Add Data Source screen opens.

3. Enter the following for the new data source:

- **Data Source Name:** The name of the data source
- **Driver Type:** Select the driver type from the drop-down list. Applicable driver types include:
 - Oracle 11g
 - Oracle 10g
 - Microsoft SQL Server 2005
- **Database Driver Class:** the driver class for the data source. Should be one of the following:
 - Oracle: "oracle.jdbc.OracleDriver."
 - MS SQL Server: "net.sourceforge.jtds.jdbc.Driver" or full classpath of JDBC driver
- **Connection String:** the connection string for the data source in one of the following formats:

For Oracle data sources:

```
jdbc:oracle:thin:@<ServerName>:<Port>:<YourDatabaseName>
```

where:

- **<ServerName>** is the machine name of the database server.
- **<Port>** is the port the database connection will use. Default is 1521.
- **<YourDatabaseName>** is the TNS service name used by the database's ODBC connection.

Note: If using Oracle 10g RAC, the connection string must specify the TNS entry for the RAC connection. See **Oracle BI Publisher Configuration for Oracle RAC** on page 2-15 in the *Oracle Utilities Energy Information Platform Installation Guide* for more information.

For MS SQL Server data sources:

```
<DRIVER_URL>://<ServerName>/<Database>
```

where:

- **<Driver_URL>** is the URL of the JDBC driver
 - **<ServerName>** is the machine name of the database server.
 - **<Database>** is the database name.
 - **User ID:** the user ID for the database connection
 - **Password:** the password for the supplied user ID
4. Click **Apply**.

The new data source appears on the JDBC tab of the Data Source screen.

Energy Information Platform Configuration

Once the Oracle BI Publisher Administrator user is created on the Oracle BI Publisher server, you must specify the URL for the server and user credentials in the LSReportMonitor.cfg.xml configuration file on each application or web server that needs to access the Oracle BI Publisher server. This includes all application servers used to generate Oracle BI Publisher reports, as well as all web servers that users will access when creating, running, and viewing reports.

In the LSReportMonitor.cfg.xml configuration file:

- The BI_PUBLISHER/URL element is used to specify the URL of the Oracle BI Publisher server.
- The BI_PUBLISHER/CONNECTSTRING element is used to specify the user credentials for the Oracle BI Publisher Administrator user you created earlier (see **Create User for Reporting Framework** on page 13-17).
- The BI_PUBLISHER/VERSION element is used to specify the version of Oracle BI Publisher being used.

The sample LSReportMonitor.cfg.xml file below shows the format of these elements.

```
<LSREPORTMONITOR>
...
  <BI_PUBLISHER>
    <URL>http://<server>:<port>/xmlpserver</URL>
    <CONNECTSTRING>UID=<userid>;PWD=<password></CONNECTSTRING>
    <VERSION>10</VERSION>
  </BI_PUBLISHER>
</LSREPORTMONITOR>
```

where:

- <server> is the machine name or IP address of the server on which the Oracle BI Publisher server is installed
- <port> is the port used to connect to the Oracle BI Publisher server
- <user_id> is the user ID for the connection to the Oracle BI Publisher server.
- <password> is the password for the supplied user ID.

Adding Oracle BI Publisher Reports to the Energy Information Platform

This section describes how users add custom Oracle BI Publisher reports to the Energy Information Platform Reporting Framework.

Adding custom reports involves the following steps:

1. **Create Report Template**
2. **Add Report Template to the Reporting Framework**
3. **Create Report Categories (optional)**
4. **Create Report Relationships in the Reporting Framework**

Note that the steps described below allow for the Reporting Framework to dynamically create ASP pages for entering report parameters. To add custom reports that use hard-coded “static” ASP pages, see **Adding Custom Reports Using Static ASP Pages** on page 13-26.

Create Report Template

The first step is to create the report you wish to add using Oracle BI Publisher (version 10.1.3.4.1 or version 11.1.1.1, available under separate license). See **Designing Oracle BI Publisher Reports for use with the Reporting Framework** on page 13-22 for more information about creating Oracle BI Publisher reports for use with the Energy Information Platform Reporting Framework.

Add Report Template to the Reporting Framework

The next step is to add the report to the Oracle Utilities Reporting Framework on the **Templates** tab of the **Report Administration** screen. Perform the following steps for each report template you wish to add:

1. Click **Add** on the Templates search results list page. A blank template screen appears.
2. Enter a **Report Name** for the report template.
3. Select the **Report Type** (BIPublisher) for the report from the drop-down list.
4. Select the report from the **Template File** drop-down list.
5. Select the **Output Format** for the report from the drop-down list. A specific output format is required for Oracle BI Publisher reports.
6. Enter an optional **Description** for the report template.
7. Set the **Allow Real Time** flag for the report template to indicate if the report can be run in real time instead of being run in a queue. This flag **must** be set to “No” when the Report Type is BIPublisher (Oracle BI Publisher).
8. Set the **Generate ASP** flag for the report template to indicate if the Report Framework should automatically generate an ASP page that prompts for report parameters when users run the report. This flag **must** be set to “Yes” when the Report Type is LSRate (Oracle Utilities Rules Language) or BIPublisher (Oracle BI Publisher).
9. Click **Save** to save the report template. Click **Reset** to clear all the fields.

Create Report Categories (optional)

The next (optional) step is to create report categories used by the Reporting Framework to group reports on the **Categories** tab of the **Report Administration** screen. Perform the following steps for each report category you wish to create:

1. Click **Add** on the Categories search results list page. A blank category screen appears.
2. Enter a name for the **Report Category**.

3. Enter a **Description** for the report category.
4. Click **Save** to save the report category. Click **Reset** to clear all the fields.

Create Report Relationships in the Reporting Framework

The last step is to create relationships between the report template and the appropriate Oracle Utilities products and/or report categories on the **Relationships** tab of the **Report Administration** screen. Perform the following steps for each report category you wish to create:

1. Click **Add** on the Relationships search results list page. A blank relationship screen appears.
2. Select the **Report Name** for the relationship from the drop-down list.
3. Select the **Product** for the relationship from the drop-down list.
4. Select an optional **Report Category** for the relationship from the drop-down list.
5. Click **Save** to save the report relationship. Click **Reset** to clear all the fields.

Designing Oracle BI Publisher Reports for use with the Reporting Framework

This section provides information useful when creating Oracle BI Publisher reports for use with the Energy Information Platform Reporting Framework, including:

- **Report Folders**
- **Report Data Sources**
- **Creating Parameters and Parameter Fields**

Refer to your Oracle BI Publisher documentation for more detailed information about creating Oracle BI Publisher reports.

Report Folders

The pre-defined Oracle BI Publisher Reports located within product-specific folders located within the “Shared Folders” folder, as follows (on the Reports tab):

Home > Shared Folders > Shared > [PRODUCT] > [REPORT]

where:

- **[PRODUCT]** is the code for the product (MDM, QM, etc.)
- **[REPORT]** is the name of the report

When creating custom reports, create all new reports using this same convention. For example, if you were to create a set of reports related to forecasting, you might create a Forecasting folder in which you would create the individual reports.

Important: The Energy Information Platform Reporting Framework administration screens can only access Oracle BI Publisher reports defined within the Shared Folders folder. Reports created in any other folder will NOT be visible via the Energy Information Platform Reporting Framework administration screens.

Report Data Sources

Oracle BI Publisher reports designed for use with the Oracle Utilities Reporting Framework should use only a single data source. Using more than one data source can produce unexpected results.

The data source defined in a Oracle BI Publisher report is dynamically substituted when the report is run via the Report Framework.

Creating Parameters

This section provides information useful when creating parameters for Oracle BI Publisher reports.

General Settings

The General Settings section of the Oracle BI Publisher Parameter screen is used to define several import properties of each parameter.

Identifier

The **Identifier** field defines the name of the parameter, and in some cases also determines the data type controls that appears on the **Enter Report Parameters** screen. Identifiers are required for all parameters. Identifier must be 32 characters or less.

Some specific applications of the Identifier field include:

- **Report Title:** To create a Report Title field, the Identifier of the parameter field MUST be RptTitle. This Name is **case-sensitive**. The value passed to the “RptTitle” parameter will be displayed in the **Report Title** column on the **View Reports** screen.
- **Lookup Fields:** To create a Lookup field, the Identifier must use the following naming convention:

<TABLE_NAME>_<COLUMN_NAME>

where:

TABLE_NAME is the name (or mnemonic) of the database table associated with the parameter field. If the <TABLE_NAME>_<COLUMN_NAME> string is more than 32 characters, you can use the table mnemonic instead of the table name.

COLUMN_NAME is the name of the column in the table associated with the parameter field.

For example, to create a report with an “Account ID” lookup parameter field, you would enter ACCOUNT_ACCOUNTID in the **Identifier** field.

Data Type

The Data Type field defines the data type for the parameter. Data Type is required for all parameters. Available data types include:

- **String:** The parameter is text value.
- **Integer:** The parameter is an integer value.
- **Boolean:** The parameter is a Yes/No or True/False value.
- **Date:** The parameter is a date and time value.
- **Float:** The parameter is a float value.

Some specific applications of the Data Type field include:

- **Report Title:** To create a Report Title field, the Data Type of the parameter field MUST be String.
- **Lookup Fields:** To create a Lookup field, the Data Type must be String.
- **Date and Date/Time Fields:** To create a Date or Date/Time field, the **Data Type** of the parameter field must be Date.

Default Value

The Default Value field defines the default value for the parameter. Default values are optional.

Parameter Type

The Parameter Type field defines the type of parameter. Parameter Type is required for all parameters. Available parameter types include:

- **Text:** this type allows the user to enter a text entry as the parameter. Text parameters use the following settings:
 - **Display Label:** The text that will appear on the Report screen for the parameter.
 - **Text Field Size:** The size of the text field, in the number of characters.
 - **Options:** Not applicable for reports run through the Energy Information Platform Reporting Framework.
- **Menu:** Not applicable for reports run through the Energy Information Platform Reporting Framework.
- **Date:** passes a date parameter. If you select a Parameter Type of Date, the Data Type automatically defaults to Date. Text parameters use the following settings:

- **Display Label:** The text that will appear on the Report screen for the parameter.
- **Date Format String:** the Java data format string for the date
- **Date From/Date To:** Not applicable for reports run through the Energy Information Platform Reporting Framework.
- **Hidden:** Not applicable for reports run through the Energy Information Platform Reporting Framework.
- **Search:** Not applicable for reports run through the Energy Information Platform Reporting Framework.

Some specific applications of the Data Type field include:

- **Report Title:** To create a Report Title field, the Parameter Type of the parameter field MUST be String.
- **Lookup Fields:** To create a Lookup field, the Parameter Type must be String:
- **Date and Date/Time Fields:** To create a Date or Date/Time field, the Parameter Type of the parameter field must be Date.

Examples

Following are some examples of parameter from the Work Queue Aging Report (WQAging)

Parameter: Report Title

Identifier: RptTitle

Data Type: String

Default Value: Aging Report

Parameter Type: Text

Display Label: Report Title

Text Field Size: 64

Parameter: Type Code

Identifier: LSWQTYPE_WQTYPECODE

Data Type: String

Default Value:

Parameter Type: Text

Display Label: Type Code

Text Field Size:

Parameter: Start Time

Identifier: STARTTIME

Data Type: Date

Default Value:

Parameter Type: Date

Display Label: Start Time

Date Format String: MM/dd/yyyy hh:mm:ss aa

Adding Custom Crystal Reports to the Oracle Utilities Energy Information Platform

This section describes how users add custom Crystal reports to the Energy Information Platform Reporting Framework.

Adding custom reports involves the following steps:

1. **Create Report Template**
2. **Add Report Template to the Reporting Framework**
3. **Create Report Categories (optional)**
4. **Create Report Relationships in the Reporting Framework**

Note that the steps described below allow for the Reporting Framework to dynamically create ASP pages for entering report parameters. To add custom reports that use hard-coded “static” ASP pages, see **Adding Custom Reports Using Static ASP Pages** on page 13-26.

Create Report Template

The first step is to create the report template (*.rpt file) for the report you wish to add using the Crystal Reports Report Designer, version 9.0 or later (available under separate license). See **Designing Crystal Reports for use with the Reporting Framework** on page 13-28 for more information about creating Crystal Reports templates for use with the Energy Information Platform Reporting Framework.

Add Report Template to the Reporting Framework

The next step is to add the report template to the Oracle Utilities Reporting Framework on the **Templates** tab of the **Report Administration** screen. Perform the following steps for each report template you wish to add:

1. Click **Add** on the Templates search results list page. A blank template screen appears.
2. Enter a **Report Name** for the report template.
3. Select the **Report Type** for the report template from the drop-down list.
4. Enter a **Template File** for the report template. This is the Crystal Reports template (*.rpt) file used for the report. To add the template file, click the **Browse** button and use the **Choose File** dialog that opens to locate and select the updated *.rpt file.
5. Enter an optional **Description** for the report template.
6. Set the **Allow Real Time** flag for the report template to indicate if the report can be run in real time instead of being run in a queue.
7. Set the **Generate ASP** flag for the report template to indicate if the Report Framework should automatically generate an ASP page that prompts for report parameters when users run the report.
8. Click **Save** to save the report template. Click **Reset** to clear all the fields.

Create Report Categories (optional)

The next (optional) step is to create report categories used by the Reporting Framework to group reports on the **Categories** tab of the **Report Administration** screen. Perform the following steps for each report category you wish to create:

1. Click **Add** on the Categories search results list page. A blank category screen appears.
2. Enter a name for the **Report Category**.
3. Enter a **Description** for the report category.

4. Click **Save** to save the report category. Click **Reset** to clear all the fields.

Create Report Relationships in the Reporting Framework

The last step is to create relationships between the report template and the appropriate Oracle Utilities products and/or report categories on the **Relationships** tab of the **Report Administration** screen. Perform the following steps for each report category you wish to create:

1. Click **Add** on the Relationships search results list page. A blank relationship screen appears.
2. Select the **Report Name** for the relationship from the drop-down list.
3. Select the **Product** for the relationship from the drop-down list.
4. Select an optional **Report Category** for the relationship from the drop-down list.
5. Click **Save** to save the report relationship. Click **Reset** to clear all the fields.

Adding Custom Reports Using Static ASP Pages

The above steps assume that the Energy Information Platform Reporting Framework will create dynamic ASP pages for entering report parameters. It is also possible to add custom reports that use custom-coded “static” ASP pages for entering report parameters.

Adding custom reports that use hard-coded ASP pages involves the following steps:

1. **Create Report Template**
2. **Create ASP Page for the Report**
3. **Add Report Template to the Reporting Framework**
4. **Create Report Relationships in the Reporting Framework**

Create Report Template

The first step is to create the report template (*.rpt file) for the report you wish to add using the Crystal Reports Report Designer, version 9.0 or later (available under separate license). See **Designing Crystal Reports for use with the Reporting Framework** on page 13-28 for more information about creating Crystal Reports templates for use with the Oracle Utilities Reporting Framework.

For example, if you were creating a report that lists customers past billing information, you might call the report template “**CustBillHist.rpt**”.

Create ASP Page for the Report

The next step is to create the ASP page that allows users to enter parameters for the report. The name of this page should be **<TEMPLATE_NAME>.asp**, where **<TEMPLATE_NAME>** is the name of the report template created in the first step.

To continue the example from the previous step, the filename of the ASP page for your customers billing history report would be “**CustBillHist.asp**”.

Copy this file into the **C:\LODESTAR\Web\LSRF\ReportTemplates** directory on the web server. Report parameters ASP pages **MUST** be located in this directory in order for the Reporting Framework to use them properly.

Passing Report Parameters to the Reporting Framework

Hard-coded ASP pages must be designed properly to pass report parameters to the Reporting Framework in a specific XML format required to generate Crystal Reports. This format is as follows:

```
<CRParameters>
  <CRParameter ParameterFieldName="Param1" CurrentValue="Value1" />
  <CRParameter ParameterFieldName="Param2" CurrentValue="Value2" />
  <CRParameter ParameterFieldName="Param3" CurrentValue="Value3" />
</CRParameters>
```

where:

- **ParameterFieldName** is the name of the parameter.
- **CurrentValue** is the value the user assigns to the parameter

The following sample ASP pages designed to work with Crystal Reports in the Reporting Framework can be found in the **C:\LODESTAR\Web\wq** directory:

- AgingReport.asp (for the Work Queue Aging report and Work Queue Open Items report)
- ProdReport.asp (for the Work Queue Productivity report)

Add Report Template to the Reporting Framework

The next step is to add the report template to the Energy Information Platform Reporting Framework on the **Templates** tab of the **Report Administration** screen. Perform the following steps for each report template you wish to add:

1. Click **Add** on the Templates search results list page. A blank template screen appears.
2. Enter a **Report Name** for the report template. The Report Name **MUST** be identical to the name of the ASP file created in the previous step (with the “.asp” removed). To continue the example from the previous step, the Report Name for your customers billing history report would be “**CustBillHist**”.
3. Select the **Report Type** for the report template from the drop-down list.
4. Enter a **Template File** for the report template. This is the Crystal Reports template (*.rpt) file used for the report. To add the template file, click the **Browse** button and use the **Choose File** dialog that opens to locate and select the updated *.rpt file.
5. Enter an optional **Description** for the report template.
6. Set the **Allow Real Tim** flag for the report template to indicate if the report can be run in real time instead of being run in a queue.
7. Set the **Generate ASP** flag for the report template to “No” to indicate that the Report Framework should **NOT** automatically generate an ASP page that prompts for report parameters when users run the report.
8. Click **Save** to save the report template. Click **Reset** to clear all the fields.

Create Report Relationships in the Reporting Framework

The last step is to create relationships between the report template and the appropriate Oracle Utilities products and/or report categories on the **Relationships** tab of the **Report Administration** screen. Perform the following steps for each report category you wish to create:

1. Click **Add** on the Relationships search results list page. A blank relationship screen appears.
2. Select the **Report Name** for the relationship from the drop-down list.
3. Select the **Product** for the relationship from the drop-down list.
4. Select an optional **Report Category** for the relationship from the drop-down list.
5. Click **Save** to save the report relationship. Click **Reset** to clear all the fields.

Designing Crystal Reports for use with the Reporting Framework

This section provides information useful when creating Crystal Reports for use with the Energy Information Platform Reporting Framework, including:

- **Report Data Sources**
- **Creating Parameters and Parameter Fields**

Refer to your Crystal Reports documentation for more detailed information about creating Crystal Reports.

Report Data Sources

Crystal Reports designed for use with the Oracle Utilities Reporting Framework should use only a single data source. Using more than one data source can produce unexpected results.

The data source defined in a Crystal Report template is dynamically substituted when the report is run via the Report Framework.

Creating Parameters and Parameter Fields

When creating Crystal Reports that require the user to input parameters, you must create parameter fields including a name, label, and value type for each.

Parameter Field Names

The **Name** field on the **Create Parameter Field** (or **Edit Parameter Field**) dialog contains the specific column in the database associated with a parameter field. The **Name** field also determines the data type controls that appears on the **Enter Report Parameters** screen.

Note: Parameter field names should be in UPPER case.

Report Title

To create a Report Title field, the Name of the parameter field **MUST** be RptTitle. This Name is **case-sensitive**. The value passed to the “RptTitle” parameter will be displayed in the **Report Title** column on the **View Reports** screen.

Lookup Fields

To create a Lookup field, the Name must use the following naming convention:

<TABLE_NAME>.<COLUMN_NAME>

where:

- TABLE_NAME is the name of the database table associated with the parameter field.
- COLUMN_NAME is the name of the column in the table associated with the parameter field.

For example, to create a report with an “Account ID” lookup parameter field, you would enter ACCOUNT.ACCOUNTID in the **Name** field.

Date and Date/Time Fields

To create a Date or Date/Time field, the Value Type of the parameter field must be either Date or DateTime. See **Parameter Field Value Types** on page 13-29 for more information.

Currency Fields

To create a currency field, the name of the field must contain the word CURRENCY, and the Value Type of the parameter field must be Currency. See **Parameter Field Value Types** on page 13-29 for more information.

For example, to create a report with an “Amount” currency parameter field, you could enter AMOUNT_CURRENCY in the **Name** field.

Parameter Field Labels

The parameter field labels that appear on the **Enter Report Parameters** screen are based on the text entered in the **Prompting text** field on the **Create Parameter Field** (or **Edit Parameter Field**) dialog. For example, a report with an “Account ID” parameter field should have enter “Account ID” in the **Prompting text** field.

Parameter Field Value Types

The data type associated with a parameter field is determined by the **Value type** drop-down list on the **Create Parameter Field** (or **Edit Parameter Field**) dialog. Crystal Reports parameter fields used in the Oracle Utilities Reporting Framework can be of one of the following value types:

- **Boolean:** The parameter is a Yes/No or True/False value.
- **Currency:** The parameter is a currency amount value.
- **Date:** The parameter is a date value.
- **DateTime:** The parameter includes both date and time values.
- **Number:** The parameter is a numeric value.
- **String:** The parameter is text value.

For example, a report with an “Account ID” parameter field would have a **Value Type** of “String”.

Adding Oracle Utilities Rules Language Rate Schedules to the Energy Information Platform

The Reporting Framework can also be used to execute and run rate schedules written using the Oracle Utilities Rules Language. This section describes how users add Rules Language rate schedules to the Energy Information Platform Reporting Framework, including:

- **Adding Rate Schedules**
- **Designing Rate Schedules for use with the Reporting Framework**

Adding Rate Schedules

Adding a rate schedule involves the following steps:

1. **Create Rate Schedule**
2. **Create Report Template**
3. **Add Report Template to the Reporting Framework**
4. **Create Report Categories (optional)**
5. **Create Report Relationships in the Reporting Framework**

Note that the steps described below allow for the Reporting Framework to dynamically create ASP pages for entering rate schedule parameters.

Create Rate Schedule

The first step is to create the rate schedule you wish to run using the Oracle Utilities Rules Language. See **Designing Rate Schedules for use with the Reporting Framework** on page 13-31 for more information about creating rate schedules for use with the Oracle Utilities Reporting Framework.

Create Report Template

The next step is to create an XML schema (*.xsd file) that defines the parameters used by the rate schedule you wish to run. See **Designing Rate Schedules for use with the Reporting Framework** on page 13-31 for more information about creating templates used with rate schedules and the Oracle Utilities Reporting Framework.

Add Report Template to the Reporting Framework

The next step is to add the report template to the Energy Information Platform Reporting Framework on the **Templates** tab of the **Report Administration** screen. Perform the following steps for each report template you wish to add:

1. Click **Add** on the Templates search results list page. A blank template screen appears.
2. Enter a **Report Name** for the report template.
3. Select the **Report Type** for the report template from the drop-down list.
4. Enter a **Template File** for the report template. This is the XML schema (*.xsd) file used for the report. To add the template file, click the **Browse** button and use the **Choose File** dialog that opens to locate and select the updated *.xsd file.
5. Enter an optional **Description** for the report template.
6. Set the **Allow Real Tim** flag for the report template to indicate if the report can be run in real time instead of being run in a queue.
7. Set the **Generate ASP** flag for the report template to indicate if the Report Framework should automatically generate an ASP page that prompts for report parameters when users run the report.

- Click **Save** to save the report template. Click **Reset** to clear all the fields.

Create Report Categories (optional)

The next (optional) step is to create report categories used by the Reporting Framework to group reports on the **Categories** tab of the **Report Administration** screen. Perform the following steps for each report category you wish to create:

- Click **Add** on the Categories search results list page. A blank category screen appears.
- Enter a name for the **Report Category**.
- Enter a **Description** for the report category.
- Click **Save** to save the report category. Click **Reset** to clear all the fields.

Create Report Relationships in the Reporting Framework

The last step is to create relationships between the report template and the appropriate Oracle Utilities products and/or report categories on the **Relationships** tab of the **Report Administration** screen. Perform the following steps for each report category you wish to create:

- Click **Add** on the Relationships search results list page. A blank relationship screen appears.
- Select the **Report Name** for the relationship from the drop-down list.
- Select the **Product** for the relationship from the drop-down list.
- Select an optional **Report Category** for the relationship from the drop-down list.
- Click **Save** to save the report relationship. Click **Reset** to clear all the fields.

Designing Rate Schedules for use with the Reporting Framework

This section provides information useful when creating rate schedules for use with the Energy Information Platform Reporting Framework, including:

- **Creating Rate Schedules**
- **Creating Templates (*.XSD Files)**

Creating Rate Schedules

Any rate schedule can be run from the Reporting Framework. When the rate schedule is executed, the values of the identifiers used in the rate schedule are passed from the Energy Information Platform to the Rules Language.

The following example rate schedule (OPEN_WQ_ITEM) allows users to open work queue items based on input parameters.

```
//Opens a Work Queue Item based on inputs from Energy Information
Platform user interface
//
//Set up Work Queue Item Attributes
WQ.WQTYPECODE = WQTYPECODE;
WQ.WQQUEUECODE = WQQUEUECODE;
WQ.ACCOUNTID = ACCOUNTID;
WQ.PRODUCTCODE = PRODUCTCODE;
WQ.ASSIGNEDTOUSERID = ASSIGNEDTOUSERID;
WQ.OPENEDNOTE = OPENEDNOTE;
//
//Open WQ Item
OPEN_WQ = WQ_OPEN(WQ);
```

Creating Templates (*.XSD Files)

Once the rate schedule(s) you wish to run using the Reporting Framework has been created, the next step is to create a template (*.xsd) file that defines the parameters used by the rate schedule.

The following illustrates the format of the XSD file used to define input parameters.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:ls="http://
www.lodestarcorp.com" attributeFormDefault="unqualified"
elementFormDefault="qualified">
  <xs:element name="Call">
    <xs:complexType>
      <xs:all>
        <xs:element name="OPCOCODE">_<JURISCODE>_<RATE_FORM_CODE">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="AutoSave" type="xs:boolean" ls:caption="Auto
Save"/>
              <xs:element name="<PARAMETER>" type="<PARAM_TYPE>"
ls:entity="<TABLE_NAME>" ls:isUid="<UID>" ls:custom="<FILTER>"
ls:caption="<FIELD_NAME>" ls:isRequired="<REQ>" default="<DEFAULT>" />
            </xs:sequence>
            <xs:attribute name="RATE_CODE" fixed="<RATE_FORM_CODE"/>
            <xs:attribute name="OPERATING_COMPANY" fixed="<OPCOCODE"/>
            <xs:attribute name="JURISDICTION" fixed="<JURISCODE"/>
            <xs:attribute name="VERSION" fixed="<VERSIONNO"/>
          </xs:complexType>
        </xs:element>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

where:

- **<OPCOCODE>** is the Operating Company code associated with the rate schedule.
- **<JURISCODE>** is the Jurisdiction code associated with the rate schedule.
- **<RATE_FORM_CODE>** is the Rate Form code of the rate schedule.
- **<VERSIONNO>** is the (optional) Rate Form Version Number of the rate schedule to execute. If not specified, the Current version of the rate schedule is executed.
- **<PARAMETER>** is the identifier used in the rate schedule. When defining a lookup, the PARAMETER name MUST be the same name as the column from which the value is to be taken. You can include any number of parameters.

The optional “AutoSave” parameter can be used to display an “Auto Save” checkbox on the Enter Report Parameters screen. If the checkbox is checked, the rate schedules saves all database changes to the database. If the checkbox is unchecked, all database changes are rolled back at the end of the execution of the rate schedule. This parameter can be used to execute Rules Language reports without saving to the database (for testing). An example of this parameter is as follows:

```
<xs:element name="AutoSave" type="xs:boolean" ls:caption="Auto Save"/>
```

- **<PARAM_TYPE>** is the type of parameter. Can be one of the following:
 - **Boolean** (xs:boolean): Defines a parameter that returns “true” or “false”. Displays a checkbox on the user interface.
 - **Date** (xs:date): Defines a date parameter. Displays a Date control on the user interface. There are two specific Date (or Datetime) parameters that can be used:
 - **INPUTSTARTDATE**: Used to set the “Bill Start” time for the Rules Language calculations. If not provided, the Bill Start is set to a default value.

- **INPUTSTOPDATE**: Used to set the “Bill Stop” time for the Rules Language calculations. If not provided, the Bill Start is set to a default value.
- **Datetime** (xs:dateTime): Defines a datetime parameter. Displays a DateTime control on the user interface.
- **Double Precision** (xs:double): Defines a float (double precision) parameter. Supports up to 2 demical places. Displays a text field on the user interface.
- **File Name** (ls:file): Defines a string parameter containing a supplied path and file name. Displays a text field on the user interface.
- **Month** (xs:MonthYear): Defines a month parameter, in the form of MM/YYYY. This is used to supply Bill Month parameters. There is one specific Month parameter that can be used:
 - **INPUTMONTHYEAR**: Used to set the “Bill Month” for the Rules Language calculations. If not provided, the Bill Month is set to a default value.
- **Long Integer** (xs:long): Defines a long integer parameter. Displays a text field on the user interface.
- **Lookup** (ls:lookup): Defines a lookup parameter. Displays a Lookup control on the user interface.
- **Value** (xs:string): Defines a string parameter. Displays a text field on the user interface.
- **Drop-down List** (ls:<NAME>): Defines a drop-down list parameter, where <NAME> is the name of the drop-down list. The drop-down list is further specified within a <xs:simpleType> element, and individual values available from the drop-down list are specified in <xs:enumeration> elements, as follows:

```
<xs:element name="SCENARIO" type="ls:<NAME>" ls:isRequired="true"
ls:caption="Scenario" default="Base"/>
<xs:simpleType name="ls:<NAME>">
  <xs:restriction>
    <xs:enumeration value="<VALUE_1>"></xs:enumeration>
    <xs:enumeration value="<VALUE_2>"></xs:enumeration>
    <xs:enumeration value="<VALUE_3>"></xs:enumeration>
  </xs:restriction>
</xs:simpleType>
```

- **<TABLE_NAME>** (ls:entity) is the name of the table in the Oracle Utilities Data Repository that is used for the look up. Used with lookups only.
- **<UID>** (ls:isUid) should be "true" when a parameter is a UID (unique identifier), such as UIDACCOUNT. Used with UID lookups only. For example:

```
<xs:element name="UIDACCOUNT" type="ls:lookup" ls:isUid="true"
ls:entity="ACCOUNT" ls:caption="Account Id" ls:isRequired="true"/>
```

- **<FIELD_NAME>** (ls:caption) is the name for the parameter field that will appear on the user interface.
- **<FILTER>** (ls:custom) is used to filter the available records for lookup parameters. Uses the following syntax:

```
ls:custom="{ 'O_<COLUMN_NAME>': '<OPERAND>', 'X_<COLUMN_NAME>': '<VALUE>' }
```

where

- **O_<COLUMN_NAME>**: is the column name to be filtered.

- **<OPERAND>** is the operand used to filter the column specified by **<COLUMN_NAME>**. This can be one of the following:

Operand	Explanation
EQ	The column value is Equal to the “X_” parameter.
NE	The column value is Not Equal to the “X_” parameter.
LT	The column value is Less Than the “X_” parameter.
LE	The column value is Less Than or Equal to the “X_” parameter.
GT	The column value is Greater Than the “X_” parameter.
GE	The column value is Greater than or Equal to the “X_” parameter.
LIKE	The column value matches the “X_” parameter.
NOTLIKE	The column value does not match the “X_” parameter.
SOUNDX	The column value is similar to the “X_” parameter.
NOTSONDX	The column value is not similar to the “X_” parameter.
NULL	The column is Null .
NOTNULL	The column is Not Null .

For example, to set the operand for the Jurisdiction (JURISCODE) column to “equals” (EQ), you would use the following:

```
ls:custom="{ 'O_JURISCODE': 'EQ', 'X_<COLUMN_NAME>': '<VALUE>' }"
```

- **X_<COLUMN_NAME>**: is the column name to be filtered.
- **<VALUE>**: value of the column specified by **<COLUMN_NAME>**. For example to set the value of the Jurisdiction (JURISCODE) column to “LODESTAR”, you would use the following:

```
ls:custom="{ 'O_JURISCODE': 'EQ', 'X_JURISCODE': 'LODESTAR' }"
```

- **<REQ>** (ls:isRequired) specifies whether or not the parameter is required. Can be one of the following:
 - **Required:** ls:isRequired="true"
 - **Optional:** ls:isRequired="false"
- **<DEFAULT>** specifies the default value for the parameter.

The following default values can be used with Date and Datetime parameters:

- **daystart:** Sets STARTTIME parameters to the start of the current day (XX/XX/XXXX 00:00:00).
- **dayend:** Sets STOPTIME parameters to the end of the current day (XX/XX/XXXX 23:59:59).
- **monthstart:** Sets STARTTIME parameters to the start of the first day of the current month (XX/XX/XXXX 00:00:00).

- **monthend:** Sets STOPTIME parameters to the end of the last day of the current month (XX/XX/XXXX 00:00:00).
- **yearstart:** Sets STARTTIME parameters to the start of the first day of the current year (XX/XX/XXXX 00:00:00).
- **yearstop:** Sets STOPTIME parameters to the end of the last day of the current year (XX/XX/XXXX 00:00:00).
- **daycurrent:** Sets Date or Datetime parameters to the current date and time.

Example

The following XSD file defines the input parameters used by the example rate schedule above (OPEN_WQ_ITEM).

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:ls="http://
www.lodestarcorp.com" attributeFormDefault="unqualified"
elementFormDefault="qualified">
  <xs:element name="Call">
    <xs:complexType>
      <xs:all>
        <xs:element name="LODESTAR_LODESTAR_OPEN_WQ_ITEM">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="WQTYPECODE" type="ls:lookup"
ls:entity="LSWQTYPE" ls:caption="Work Queue Type" ls:isRequired="true"/>
              <xs:element name="WQQUEUECODE" type="ls:lookup"
ls:entity="LSWQQUEUE" ls:caption="Work Queue" ls:isRequired="true"/>
              <xs:element name="ACCOUNTID" type="ls:lookup" ls:entity="ACCOUNT"
ls:caption="Account ID" ls:isRequired="false" default="800001"/>
              <xs:element name="PRODUCTCODE" type="ls:lookup"
ls:entity="LSPRODUCT" ls:caption="Product" ls:isRequired="true"/>
              <xs:element name="ASSIGNEDTOUSERID" type="xs:string"
ls:caption="User ID" ls:isRequired="true"/>
              <xs:element name="OPENEDNOTE" type="xs:string" ls:caption="Note"
ls:isRequired="false"/>
            </xs:sequence>
            <xs:attribute name="RATE_CODE" fixed="OPEN_WQ_ITEM"/>
            <xs:attribute name="OPERATING_COMPANY" fixed="LODESTAR"/>
            <xs:attribute name="JURISDICTION" fixed="LODESTAR"/>
          </xs:complexType>
        </xs:element>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

When selected via the **Run Reports** screen, the above file would create the following **Enter Report Parameters** screen:

When the user clicks **Run**, the rate schedule specified in the *.XSD file would be executed, and would generate a Work Queue based on the supplied parameters.

Reporting on Interval Data

The Energy Information Platform Reporting Framework also allows users the capability to create Crystal Reports based on interval data stored in the Oracle Utilities Data Repository. This section describes the steps involved in reporting on interval data, including:

- **Creating Crystal Reports Templates**
- **Converting Interval Data**
- **Interval Data Reporting Tables**

Creating Crystal Reports Templates

Creating Crystal Reports templates for reporting on interval data is very similar to creating templates for any sort of reporting using the Oracle Utilities Data Repository, and involves the following steps:

- **Create Crystal Reports Template**
- **Add Crystal Reports Template to Reporting Framework**

Create Crystal Reports Template

The first step is to create a Crystal Reports template that reports on data in the Interval Data Reporting tables in the Oracle Utilities Data Repository. See **Designing Crystal Reports for use with the Reporting Framework** on page 13-28 for more information about creating Crystal Reports templates. See **Interval Data Reporting Tables** on page 13-38 for more information about the interval data reporting tables.

Add Crystal Reports Template to Reporting Framework

The next step is to add the Crystal Reports template to the Reporting Framework using the Report Administration user interface. See **Adding Custom Crystal Reports to the Oracle Utilities Energy Information Platform** on page 13-25 more information.

Converting Interval Data

Converting interval data involves copying the interval data you wish to report on into the interval data reporting tables. This can be accomplished using the ConvertIntervalData report available on the Tools and Utilities Run Reports screen of the Energy Information Platform. This report uses the Oracle Utilities Rules Language to copy the specified interval data into the reporting tables. See **ConvertIntervalData Processing** on page 13-37 for more information about this report.


How to convert interval data for reporting:

1. Select **Run Reports** on the Tools and Utilities menu of the Energy Information Platform user interface. The Run Reports screen opens.
2. Click the Report Name link for the **ConvertIntervalData** report. The Enter Report Parameters screen opens.
3. Select the Recorder ID for the interval data you wish to convert by clicking the **Browse** button (...). This opens a dialog window that lists all the recorder IDs available. If there is more than one page of records, you can move between pages using the ◀ and ▶ buttons. Use the ⏮ button to return the first page, and the ⏭ button to navigate to the last page. To change the number of results that appears on each page, enter the number in the Rows field and press the Enter key.

To search for a specific record, click **Search**. The dialog displays a search screen.

Enter the appropriate parameters for your search and click **Search**. The dialog will display a list of records that match the search criteria.

Click the [...] link to select the appropriate record. The Recorder ID field is automatically populated.

4. Enter the channel number for the interval data you wish to convert in the **Channel Number** field.
5. Enter **Start Time** and **Stop Time** of the desired date range by either typing the date and time manually, or clicking the arrow button  to the right of the fields, selecting the date using the calendar control, and entering the time manually.
6. Click **Run** to run the report. Click **Schedule Report** to schedule the report using the Schedule Process screen (see **Scheduling Reports** on page 7-75 in the *Oracle Utilities Energy Information Platform User's Guide* for more information). When the report is complete, the report opens (see **Viewing Oracle Utilities Rules Language Reports** on page 7-79 in the *Oracle Utilities Energy Information Platform User's Guide* for more information about viewing Oracle Utilities Rules Language reports). In addition, the report is also listed on the **View Reports** screen.

Once the interval data has been copied into the interval data reporting tables, users can run reports on the data using the appropriate Crystal Reports template.

ConvertIntervalData Processing

The ConvertIntervalData report uses the Oracle Utilities Rules Language to copy specified interval data into the Interval Data Reporting tables. See **Adding Oracle Utilities Rules Language Rate Schedules to the Energy Information Platform** on page 13-30 for more information about how the Reporting Framework uses the Oracle Utilities Rules Language.

Report Template (*.XSD File)

The following XSD file defines the input parameters used by the ConvertIntervalData report.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:ls="http://
www.lodestarcorp.com" attributeFormDefault="unqualified"
elementFormDefault="qualified">
  <xs:element name="Call">
    <xs:complexType>
      <xs:all>
        <xs:element name="LODESTAR_LODESTAR_LSTU_CONVERT_INTERVAL_DATA">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="RECORDERID" type="ls:lookup"
ls:entity="RECORDER" ls:caption="Recorder ID" ls:isRequired="true"/>
              <xs:element name="CHANNELNUM" type="xs:string"
ls:caption="Channel Number" ls:isRequired="true"/>
              <xs:element name="STARTTIME" type="xs:dateTime" ls:caption="Start
Time" ls:isRequired="true" default="daystart"/>
              <xs:element name="STOPTIME" type="xs:dateTime" ls:caption="Stop
Time" ls:isRequired="true" default="daystopt"/>
            </xs:sequence>
            <xs:attribute name="RATE_CODE" fixed="LSTU_CONVERT_INTERVAL_DATA"/>
            <xs:attribute name="OPERATING_COMPANY" fixed="LODESTAR"/>
            <xs:attribute name="JURISDICTION" fixed="LODESTAR"/>
          </xs:complexType>
        </xs:element>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Rules Language

The ConvertIntervalData report uses the following Rules Language to copy the data into the interval data reporting tables:

```
LABEL RECORDERID "Recorder ID";
LABEL CHANNELNUM "Channel Number";
LABEL STARTTIME "Start Time";
LABEL STOPTIME "Stop Time";
//
RECORDER_CHANNEL = RECORDERID + "," + CHANNELNUM;
INTERVAL_DATA_TABLESET = "RDB/LSRFINTDHEADER;" + RECORDER_CHANNEL;
HNDL = INTDLOADDATES(RECORDER_CHANNEL , STARTTIME , STOPTIME);
SAVE HNDL TO STAGING INTERVAL_DATA_TABLESET;
```

Interval Data Reporting Tables

Reporting on interval data requires that the interval data be in the Interval Data Reporting tables: the LSRDINTDHEADER table, and the LSRFINTDVALUES table. These tables store interval data as individual cuts, similar to the way interval data is stored in the LSChannelcut Header and LSChannel Cut Data tables, but in a format that is accessible by Crystal Reports. For each cut stored in these tables, there is one record in the LSRFINTDHEADER table, and one record for each interval in the cut in the LSRFINTDVALUES table.

LSRFINTDHEADER Table

Records in the LSRFINTDHEADER table contain header information for each interval data cut, and include the following information:

- **UIDINTDHEADER:** The unique identifier for the cut in the table. This is also the primary key of the record.
- **RECORDER:** The Recorder ID of the cut.
- **CHANNEL:** The channel number of the cut.
- **SPI:** The Seconds-per-interval (SPI) of the cut.
- **UOMCODE:** The Unit-of-Measure (UOM) code of the cut.
- **DSTPARTICIPANT:** The cut's DST Participant flag.
- **ORIGIN:** The cut's Origin flag.
- **TZSTDNAME:** The Time Zone Standard Name for the cut.

LSRFINTDVALUES Table

Records in the LSRFINTDVALUES table store individual interval values for each interval data cut. Each record in this table is the child of a corresponding record in the LSRFINTDHEADER table, and includes the following information:

- **UIDINTDHEADER:** The unique identifier of the cut, from the LSRFINTDHEADER table.
- **DATETIME:** The interval timestamp.
- **VALUE:** The interval value.
- **STATUS:** The interval status code.

Chapter 14

Troubleshooting

This chapter describes tools available to aid in troubleshooting performance problems and other issues that may arise when installing and configuring Oracle Utilities Energy Information Platform, including:

- **Rules Language Profiling**
- **SQL Tracing**
- **Oracle Utilities Energy Information Platform Diagnostics**
- **Oracle Utilities Energy Information Platform Adapter Logging**

Rules Language Profiling

Rules Language profiling allows administrators to create a “Code Profile” that lists how long (in milliseconds) it takes the Rules Language to execute each line of code in a profiled rate schedule. This can aid in troubleshooting Rules Language performance issues and problems that may arise during configuration and testing of Oracle Utilities software. Rules Language profiling can be performed in one of two ways:

- **Running Trial Calculations**
- **Within an Application**

Important Note

Rules Language profiling has a negative impact on performance, and should ONLY be used when troubleshooting and/or debugging; Rules Language profiling should be DISABLED when running Oracle Utilities software in a production environment.

Running Trial Calculations

The Trial Calculation module (see **Trial Calculations** on page 10-2 in the *Data Manager User's Guide*) includes the ability to create a code profile when running an analysis in Single-Step mode.

How to create a code profile when using Trial Calculation:

1. Open a Single-Step analysis in the Trial Calculation module in either Data Manager or C/S Oracle Utilities Billing Component.
 - a. Select **Analysis->Trial Calculation** in Data Manager.
 - b. Select the Customer/Account, Bill Period, and Rate Schedule as appropriate.
 - b. Set Options and/or Advanced Options as appropriate.
 - b. Click Single-Step to run the analysis in Single-Step mode.

See **How to Run a Trial Calculation:** on page 10-2 and **How to run the Single Step Analysis** on page 10-4 in the *Data Manager User's Guide* and for more details about running a trial calculation in Single-Step mode.

2. When the windows for the Single-Step mode open, select **Single-Step->Run** (or press **F5**).
3. After the analysis is complete, select **Single-Step->Display Profile** (or press **Shift-F6**)/

The code profile will open in a new report window titled “**Performance Statistics**”. See **Sample Profile Output** on page 14-4 for a sample of the contents of this window.

When creating code profiles using the Trial Calculation module, the code profile generated can include a memory snapshot that displays the memory usage of the Data Manager application by including the MEM_USAGE_DISPLAY_RLP parameter in the LODESTAR.CFG configuration file. See **LODESTAR.CFG** on page 2-2 for more information about this parameter.

Within an Application

You can also obtain a Rules Language code profile when executing Rules Language within any Oracle Utilities application, including RUNRS.EXE and the Oracle Utilities Rules Language interface, as well as within rate schedules executed by Oracle Utilities products such as Oracle Utilities Energy Information Platform, Oracle Utilities Billing Component, Oracle Utilities Quotations Management, or Oracle Utilities Load Profiling and Settlement.

Rules Language profiling from within an application can be accomplished through use of:

- **SAVE_PROFILE Function**
- **LS_SAVE_PROFILE_FILENAME Identifier**

SAVE_PROFILE Function

The SAVE_PROFILE function saves a Rules Language code profile to a specified directory/file for the rate schedule in which it's used.

See **SAVE_PROFILE Function** on page 14-3 in the *Oracle Utilities Rules Language Reference Guide* for more information about using this function.

LS_SAVE_PROFILE_FILENAME Identifier

The LS_SAVE_PROFILE_FILENAME identifier specifies the path and file name of a text file that contains the code profile of the rate schedule in which the identifier appears.

Note: This identifier takes precedence over the SAVE_PROFILE function. If this identifier is present in a rate schedule (or INCLUDED Rider or Contract) all calls to the SAVE_PROFILE function are ignored.

This identifier can be set in the following ways:

- In the rate schedule itself, in which case the identifier is used in an ASSIGNMENT statement in the rate schedule.
- Passed as an input parameter to the rate schedule. This can be accomplished through use of the “-x” parameter of RUNRS.EXE or by any other means of passing input parameters to a rate schedule.

See **Reserved Identifiers** on page 4-13 in the *Oracle Utilities Rules Language User's Guide* for more information about this identifier.

Rules Language Profiling When Processing Multiple Accounts

It's also possible (and sometimes desirable) to obtain a code profile when processing multiple customers and/or accounts. This allows you to obtain a code profile that lists overall execution time for a rate schedule for a set of customers and/or accounts, rather than just the execution time for processing a single customer or account.

Multiple Accounts Running the Same Rate Schedule

When the same rate schedule is executed for multiple accounts within a single run of an application, the profile for all the accounts will be accumulated and saved to the specified file.

For example, if processing a group of 100 accounts, the time listed for each line of Rules Language code will represent the total execution time for that line for all accounts processed.

Multiple Accounts Running Different Rate Schedules

When different rate schedules are executed for each customer and/or account being processed. Whenever a rate schedule changes, the profile information will be cleared. For example if two rate schedules are run alternatively (the first for Account “A”, the second for Account “B”, the first again for Account “C”, etc.), at the end of processing, the profile information for the last run of the two rates alone will be available.

Cases where a single rate schedule INCLUDEs or CALLs different riders will be treated as a rate schedule change. However, in these cases, the profiles will be saved with a number appended to the end of the file name. For example, suppose a rate schedule creates a code profile file called “PROFILE.TXT.” If the rate schedule is executed twice (for two accounts) and INCLUDEs or CALLs different riders during each execution, the profiles would be saved as “PROFILE.TXT” and “PROFILE_1.TXT.”

Sample Profile Output

The following is an example of a code profile for a portion of a simple billing calculation rate schedule.

Line#	milliseconds	hits	Code
1			// Saved by lprosper on 12/17/2004 15:47:26
2			//Created by: 11/06/2001 L. Prosperi
3			//Rate: RS_1 Residential Service Rate 1
4			//
5			//Load Account Data
6			//
7	0.08	1	ACCOUNT_ID = ACCOUNT.ACCOUNTID;
8	0.01	1	START = BILL_START;
9	0.01	1	STOP = BILL_STOP;
10			//
11			//Cancel Rebill Rider
12	0.00	1	INCLUDE "CANCEL_REBILL_RIDER";
13			// Saved by lprosper on 12/17/2004 15:47:10
14			/** CANCEL_REBILL_RIDER for BX Training*/
15			/* */
16	0.03	1	IF (BILL_TYPE = "CANCEL") OR (BILL_TYPE
17			= "CANCEL/REBILL")
18			THEN
19			CAN.ACCT_NO = ACCOUNT_ID;
20			CAN.KWH = KWH;
21			CAN.TOT_DUE = \$EFFECTIVE_REVENUE;
22			CAN.BILL_TYPE = BILL_TYPE;
23			CAN.BILL_START = BILL_START;
24			CAN.BILL_STOP = BILL_STOP;
25			SAVE CAN TO CIS SECTION "CANCEL";
26			END IF;
27			/* End INCLUDE "CANCEL_REBILL_RIDER";
			(Nesting Level 1) */
			//

SQL Tracing

SQL tracing allows administrators to trace (and capture in a log file) all database queries executed by Oracle Utilities applications. This can aid in troubleshooting performance issues and database-related problems that may arise during configuration and testing of Oracle Utilities software.

Important Note

SQL tracing has a negative impact on performance, and should ONLY be used when troubleshooting and/or debugging; SQL tracing should be DISABLED when running Oracle Utilities software in a production environment.

Enabling SQL Tracing

SQL tracing for all Oracle Utilities applications (including c/s applications, command line programs, and web applications) can be logged using the LSDB.CFG.XML configuration file. This file defines a log file used to capture all ODBC statements (queries) executed by all Oracle Utilities applications. The LSDB.CFG.XML file must be installed in the **C:\LODESTAR\CFG** directory.

Example:

```
<LS_SQL_CONFIG>
  <SQL_TRACE ENABLE="YES" LOG_PARAMS="YES" FILENAME="U:\SQLLOG.LOG"
  APPEND="NO"/>
</LS_SQL_CONFIG>
```

See **LSDB.CFG.XML** on page 2-26 for more information about setting up this file.

Sample SQL Tracing File

The text below is representative of the output of SQL tracing.

```
...
select distinct  TR0804.ACCOUNT.UIDACCOUNT,
TR0804.ACCOUNT.UIDCUSTOMER,  TR0804.ACCOUNT.ACCOUNTID,
TR0804.ACCOUNT.STARTTIME,  TR0804.ACCOUNT.STOPTIME,
TR0804.ACCOUNT.OPCOCODE,  TR0804.ACCOUNT.JURISCODE,
TR0804.ACCOUNT.NAME,  TR0804.ACCOUNT.SIC,
TR0804.ACCOUNT.ACCOUNTSTATUSCODE,  TR0804.ACCOUNT.REVENUECODE,
TR0804.ACCOUNT.BILLINGMODEFLAG,  TR0804.ACCOUNT.PRINTDETAIL,
TR0804.ACCOUNT.FULLDAYBILL,  TR0804.ACCOUNT.PREWINDOW,
TR0804.ACCOUNT.POSTWINDOW,  TR0804.ACCOUNT.EDI,
TR0804.ACCOUNT.REGIONCODE,  TR0804.ACCOUNT.CURRENCYCODE,
TR0804.ACCOUNT.OWNER,  TR0804.ACCOUNT.ACCOUNTTYPECODE,
TR0804.ACCOUNT.UIDPARTY,  TR0804.ACCOUNT.UIDMARKETPARTICIPANT,
TR0804.ACCOUNT.SPCLHANDLECODE,  TR0804.ACCOUNT.LSUSER,
TR0804.ACCOUNT.LSTIME from  TR0804.ACCOUNT where
TR0804.ACCOUNT.ACCOUNTID <= 'BXT01_RES5' order by
TR0804.ACCOUNT.ACCOUNTID

select  count(distinct  TR0804.ACCOUNT.UIDACCOUNT)  from
TR0804.ACCOUNT where  TR0804.ACCOUNT.ACCOUNTID <= 'BXT01_RES5'
...
```

Oracle Utilities Energy Information Platform Diagnostics

One of the most important steps in installing and configuring the Oracle Utilities Energy Information Platform is set up of the LSSECURE.CFG.XML configuration file. This file specifies the database where security information (including user authentication data such as user IDs and passwords) is stored, and is required in order for the Energy Information Platform to function properly. Errors made while setting up this file can result in the user seeing an error message when they attempt to open the Login screen for the Energy Information Platform.

This section describes some of the more common error messages encountered when attempting to login to the Energy Information Platform.

Incorrect DSN in LSSECURE.CFG.XML File

LSDB5107: Unable to select USER record(s): [Microsoft][ODBC Driver Manager]
Data source name not found and no default driver specified 0x9000226a in
(2) /lodestar/ccs/default.asp

Cause of Error: The DSN specified in the <CONNECTSTRING> element in the LSSECURE.CFG.XML file is incorrect.

Remedy: Verify the correct name of the DSN and update the LSSECURE.CFG.XML file accordingly.

Incorrect User ID or Password in LSSECURE.CFG.XML File

LSDB5107: Unable to select USER record(s): [Oracle][ODBC][Ora]ORA-01017:
invalid username/password; logon denied 0x9000226a in
(2) /lodestar/ccs/default.asp

Cause of Error: The User ID (UID) and/or Password (PWD) specified in the <CONNECTSTRING> element in the LSSECURE.CFG.XML file is incorrect.

Remedy: Verify the correct user ID and/or password and update the LSSECURE.CFG.XML file accordingly.

Incorrect Qualifier in LSSECURE.CFG.XML File

LSDB5107: Unable to select USER record(s): [Oracle][ODBC][Ora]ORA-00942:
table or view does not exist 0x9000226a in (2) /lodestar/ccs/default.asp

Cause of Error: The qualifier specified in the <QUALIFIER> element in the LSSECURE.CFG.XML file is incorrect.

Remedy: Verify the correct qualifier and update the LSSECURE.CFG.XML file accordingly.

TNS Configuration (TNSNames.ora) not correctly configured

LSDB5107: Unable to select USER record(s): [Oracle][ODBC][Ora]ORA-12154:
TNS:could not resolve service name 0x9000226a in (2) /lodestar/ccs/default.asp

Cause of Error: Either the TNS Service Name specified for the DSN (on the ODBC Driver Configuration dialog) is incorrect, or there is an error in the TNSNames.ora file for the selected TNS Service Name.

Remedy: Verify the TNSNames.ora file is correct, and select the correct TNS Service Name

Oracle Utilities Energy Information Platform Adapter Logging

The Oracle Utilities Energy Information Platform Adapter (and Oracle Utilities Transaction Management) can be configured to create log files that capture the activity of Adapter business rules and services for diagnostic and troubleshooting purposes. This section provides an overview of the log files produced by the Adapter, including:

- **Setting the Debug Level**
- **Viewing Log Files**
- **Debug Level Details and Sample Log Files**

Setting the Debug Level

The specific data logged and the amount of information captured in Adapter log files are determined by the Debug Level setting in the System Properties table. See **System Properties** on page 11-12 in the *Oracle Utilities Energy Information Platform User's Guide* for information about creating, viewing, and maintaining System Properties records.

Creating log files allows administrators to diagnose problems with Adapter services and business rules, but has a negative effect on performance. For the best performance, set this to 0 (zero). At level 1, output is sent to a log file, but not to the console. Levels 2 and 3 provide additional output in both the log file and the console. Levels 4 and 5 are considered diagnostic modes, and should not be used unless under specific instruction from a technical support specialist. **Debug Level Details and Sample Log Files** on page 14-9 for more information about the specific data captured in the log file for each Debug Level setting.

Including Runtime Service Memory Profiles

Periodic logging of the number of running Adapter Runtime Services and their individual memory profiles can be included in the Adapter/Transaction Management log files. The SERVICEPROFILINGENABLED and SERVICEPROFILINGINTERVAL attributes of the LTHM element of the LTMH.CFG.XML configuration file enable and define the parameters of this feature. See **LTMH.CFG.XML** on page 2-46 for more information.

Adapter Server and Runtime Service Network Communication

Network communication operational logging can be included in the Adapter/Transaction Management log files. By default, this information is included in the main logs as well as in separate log files (with "NETWORK" appended to the log file name), but the system can be configured to include this information only in the separate log files. The NETWORKCOMMUNICATIONENABLED and NETWORKLOGTOFILEONLY attributes of the LTHM element of the LTMH.CFG.XML configuration file enable and define the parameters of this feature. See **LTMH.CFG.XML** on page 2-46 for more information.

Viewing Log Files

Adapter log files are created in the **C:\LODESTAR\LTMH\Runtime\Log** directory on the application or web server on which the Adapter software is running.

A separate log file is created for each Runtime Service that is run on the application or web server. The name of each log file is as follows:

<SERVICE_NAME>.log

where:

- <SERVICE_NAME> is the name of the Runtime Service

Log files are appended to each time a service is executed.

All events and actions captured in each log file are time-stamped in the following format:

YYYY/MM/DD HH:MM:SS:mmm - <EVENT>

Example:

2007-05-24 13:48:16.523 - Import_CSV_Demo started.

Debug Level Details and Sample Log Files

The Debug Level on the System Properties table ranges from 0 (zero) to 5, with each higher setting providing additional detail in the log file, and displaying similar information in the Adapter Server console window (see **The Adapter Server** on page 6-76 for more information about starting and running the Adapter server). This section outlines the specific data captured by each Debug Level, and provides sample log files for each setting.

Debug Level 0

At Debug Level 0 (the lowest setting), the Adapter writes no information to the log file (the log file is empty), but displays some information on the Adapter Server console.

At this level, the Adapter console displays the following:

- Service Starting message
2007-05-24 13:48:16.523 - Import_CSV_Demo started.

Debug Level 1

At Debug Level 1, additional detail is sent to the log file, but not to the Adapter Server console. At this level, the Adapter log file contains the following:

- Service Starting message (including process ID)
2007-05-24 13:48:16.523 - Import_CSV_Demo started with process id 1580.
- Initializing Process message
2007-05-24 13:48:16.783 - Initializing process: Import_CSV_Demo
- Output Mechanism message
2007-05-24 13:48:18.366 - Created FileOutputter.
- Initialization Complete message
2007-05-24 13:48:18.366 - Initialization completed.
- Process Started message
2007-05-24 13:48:28.410 - RT_FilePortal.ProcessFile Started
=====
- Applying Business Rule message (if applicable)
2007-05-24 13:48:29.111 - Applying Business Rule.
- Error XML string (if an error occurred)
2008-10-02 11:25:22.640 - <ERROR CODE="100" DESC="BRE: Meter ID BXT00_RES10 does not exist." DATE="2008-10-02T10:25:22"></ERROR>
- Success/Failure message
2007-05-24 13:48:40.537 - The rule returned successfully.
2008-10-02 11:25:22.640 - The rule returned a failure.
- Business Exception/Rollback message (if an error occurred)
2008-10-02 11:25:22.687 - A business exception was encountered. Rolling back.

- Exception message (if an error occurred - contains details of the Java class where the exception occurred)

```
2008-10-02 11:25:22.687 - Exception:
com.lodestarcorp.portal.PortalException
```

- Error message (if an error occurred)

```
2008-10-02 11:25:22.687 - BRE: Meter ID BXT00_RES10 does not exist.
```

- Process Finished message

```
2008-10-02 11:25:23.812 - RT_FilePortal.ProcessFile Finished
=====
```

Sample Log Files - Debug Level 1

Successful:

```
2007-05-24 13:48:16.523 - Import_CSV_Demo started with process id 1580.
2007-05-24 13:48:16.783 - Initializing process: Import_CSV_Demo
2007-05-24 13:48:18.366 - Created FileOutputter.
2007-05-24 13:48:18.366 - Initialization completed.
2007-05-24 13:48:28.410 - RT_FilePortal.ProcessFile Started
=====
2007-05-24 13:48:29.111 - Applying Business Rule.
2007-05-24 13:48:40.537 - The rule returned successfully.
2007-05-24 13:48:40.557 - RT_FilePortal.ProcessFile Finished
=====
```

Error:

```
2008-10-02 11:25:01.343 - Import_CSV_Demo started with process id 1580.
2008-10-02 11:25:01.359 - Initializing process: Import_CSV_Demo
2008-10-02 11:25:01.828 - Created FileOutputter.
2008-10-02 11:25:01.843 - Initialization completed.
2008-10-02 11:25:21.843 - RT_FilePortal.ProcessFile Started
=====
2008-10-02 11:25:22.015 - Applying Business Rule.
2008-10-02 11:25:22.640 - <ERROR CODE="100" DESC="BRE: Meter ID BXT00_RES10
does not exist." DATE="2008-10-02T10:25:22"></ERROR>

2008-10-02 11:25:22.640 - The rule returned a failure.
2008-10-02 11:25:22.687 - A business exception was encountered. Rolling back.
2008-10-02 11:25:22.687 - Exception: com.lodestarcorp.portal.PortalException
2008-10-02 11:25:22.687 - BRE: Meter ID BXT00_RES10 does not exist.
2008-10-02 11:25:23.812 - RT_FilePortal.ProcessFile Finished
=====
```

Debug Level 2

At Debug Level 2, additional details are sent to both the log file and to the Adapter Server console. At this level, the Adapter log file contains the following:

- Service Starting message (including process ID)

```
2007-05-24 13:28:46.450 - Import_CSV_Demo started with process id 1580.
```

- Initializing Process message

```
2007-05-24 13:28:46.731 - Initializing process: Import_CSV_Demo
```

- Runtime Service Properties

```

2007-05-24 13:28:47.332 -
=====
=====
==
==                                PROPERTIES                                ==
=====
=====
== INPUT_DC_0                      =                                           ==
==      com.lodestarcorp.core.xml.parsers.ColumnCountCSVParser
==
== INPUT_DC_1                      =                                           ==
==      com.lodestarcorp.portal.data.XSLConverter,C:\LODESTAR\LTMH\CSV_METE
==
==      R_READS.xsl
== INPUT_DC_2                      =                                           ==
==      com.lodestarcorp.portal.data.SchemaValidator,C:\LODESTAR\LTMH\Impor
==
==      t_CSV_Schema.xsd
== OUTPUT_FILE_DIR                 = C:\Training\Output
== OUTPUT_FILE_SUFFIX              = .XML
== OUTPUT_MECHANISM                = F
== POLL_DIRECTORY                  = C:\Training\CSV
== POLL_EXTENSION                  = txt
== POLL_INTERVAL                   = 10
== RDL_MEMORY_SIZE                 = 3
== RULE_NAME                       = Import_CSV_Demo
== STORAGE_LEVEL                   = 2
== WQ_ASSIGNED_TO_USERID           = lou_p
== WQ_TYPE                         = ADAPTER
=====
=====

```

- Data Conversion Object creation message(s) (if applicable)

```

2007-05-24 13:28:47.432 - Created data conversion object INPUT_DC_0:
com.lodestarcorp.core.xml.parsers.ColumnCountCSVParser

```

- Business Rule Initialization message

```

2007-05-24 13:28:48.263 - Business Rule initialization completed.

```

- Output Mechanism message (if applicable)

```

2007-05-24 13:28:48.293 - Created FileOutputter.

```

- Initialization Complete message

```

2007-05-24 13:28:48.353 - Initialization completed.

```

- Process Started message

```

2007-05-24 13:29:08.512 - RT_FilePortal.ProcessFile Started
=====

```

- Applying Business Rule message (if applicable)

```

2007-05-24 13:29:09.283 - Applying Business Rule.

```

- Mapped values, one for each column mapped by the RDL (if applicable - RDL processing only)*

```

2007-05-24 13:29:09.934 - METERREAD.METERID = BXT00_RES1

```

- Script Parsing message (if applicable)*

```

2007-05-24 13:29:10.525 - Parse Script  JSCRIPT

```

- Error XML string (if an error occurred)*

- Success/Failure message*

2007-05-24 13:29:17.926 - The rule returned successfully.

- Business Exception/Rollback message (if an error occurred)*
- Exception message (if an error occurred - contains details of the Java class where the exception occurred)*
- Error message (if an error occurred)*
- Output creation message (if applicable)

2007-05-24 13:29:17.926 - Output created at
C:\Training\Output\out_19763.XML

- Process Finished message

2007-05-24 13:29:17.936 - RT_FilePortal.ProcessFile Finished
=====

*These lines are repeated for each record within the payload being processed.

Sample Log File - Debug Level 2

```
2007-05-24 13:28:46.450 - Import_CSV_Demo started with process id 1580.
2007-05-24 13:28:46.731 - Initializing process: Import_CSV_Demo
2007-05-24 13:28:47.332 -
=====
=
==                               PROPERTIES                               ==
=====
=
== INPUT_DC_0                    =                                           ==
==   com.lodestarcorp.core.xml.parsers.ColumnCountCSVParser                ==
== INPUT_DC_1                    =                                           ==
==   com.lodestarcorp.portal.data.XSLConverter,C:\LODESTAR\LTMH\CSV_METE    ==
==   R_READS.xml                                                           ==
== INPUT_DC_2                    =                                           ==
==   com.lodestarcorp.portal.data.SchemaValidator,C:\LODESTAR\LTMH\Impor  ==
==   t_CSV_Schema.xsd                                                      ==
== OUTPUT_FILE_DIR               = C:\Training\Output                       ==
== OUTPUT_FILE_SUFFIX           = .XML                                       ==
== OUTPUT_MECHANISM              = F                                         ==
== POLL_DIRECTORY                = C:\Training\CSV                           ==
== POLL_EXTENSION                = txt                                       ==
== POLL_INTERVAL                 = 10                                         ==
== RDL_MEMORY_SIZE               = 3                                       ==
== RULE_NAME                     = Import_CSV_Demo                       ==
== STORAGE_LEVEL                 = 2                                       ==
== WQ_ASSIGNED_TO_USERID         = lou_p                               ==
== WQ_TYPE                       = ADAPTER                               ==
=====
=
2007-05-24 13:28:47.432 - Created data conversion object INPUT_DC_0:
com.lodestarcorp.core.xml.parsers.ColumnCountCSVParser
2007-05-24 13:28:47.702 - Created data conversion object INPUT_DC_1:
com.lodestarcorp.portal.data.XSLConverter
2007-05-24 13:28:47.852 - Created data conversion object INPUT_DC_2:
com.lodestarcorp.portal.data.SchemaValidator
2007-05-24 13:28:48.263 - Business Rule initialization completed.
2007-05-24 13:28:48.293 - Created FileOutputter.
2007-05-24 13:28:48.353 - Initialization completed.
2007-05-24 13:29:08.512 - RT_FilePortal.ProcessFile Started
=====
2007-05-24 13:29:09.283 - Applying Business Rule.
2007-05-24 13:29:09.934 - METERREAD.METERID = BXT00_RES1
2007-05-24 13:29:09.934 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:29:09.934 - METERREAD.SERIALNO = 12345
2007-05-24 13:29:09.944 - METERREAD.UNINUMBER = 54321
2007-05-24 13:29:09.944 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
```

```
2007-05-24 13:29:10.525 - Parse Script  JSCRIPT
2007-05-24 13:29:10.525 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:29:10.525 - METERREAD.STOPREADING = 1500
2007-05-24 13:29:10.525 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:29:10.545 - METERREAD.METERID = BXT00_RES2
2007-05-24 13:29:10.545 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:29:10.545 - METERREAD.SERIALNO = 12345
2007-05-24 13:29:10.545 - METERREAD.UNINUMBER = 54321
2007-05-24 13:29:10.545 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:29:10.545 - Parse Script  JSCRIPT
2007-05-24 13:29:10.545 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:29:10.555 - METERREAD.STOPREADING = 1500
2007-05-24 13:29:10.555 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:29:10.565 - METERREAD.METERID = BXT00_RES3
2007-05-24 13:29:10.565 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:29:10.575 - METERREAD.SERIALNO = 12345
2007-05-24 13:29:10.575 - METERREAD.UNINUMBER = 54321
2007-05-24 13:29:10.575 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:29:10.575 - Parse Script  JSCRIPT
2007-05-24 13:29:10.575 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:29:10.575 - METERREAD.STOPREADING = 1500
2007-05-24 13:29:10.575 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:29:10.595 - METERREAD.METERID = BXT00_RES4
2007-05-24 13:29:10.595 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:29:10.595 - METERREAD.SERIALNO = 12345
2007-05-24 13:29:10.595 - METERREAD.UNINUMBER = 54321
2007-05-24 13:29:10.595 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:29:10.595 - Parse Script  JSCRIPT
2007-05-24 13:29:10.665 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:29:10.665 - METERREAD.STOPREADING = 1500
2007-05-24 13:29:10.665 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:29:10.675 - METERREAD.METERID = BXT00_RES5
2007-05-24 13:29:10.775 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:29:10.795 - METERREAD.SERIALNO = 12345
2007-05-24 13:29:10.795 - METERREAD.UNINUMBER = 54321
2007-05-24 13:29:10.795 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:29:10.795 - Parse Script  JSCRIPT
2007-05-24 13:29:10.795 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:29:10.805 - METERREAD.STOPREADING = 1500
2007-05-24 13:29:10.805 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:29:10.825 - METERREAD.METERID = BXT00_GS1
2007-05-24 13:29:10.825 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:29:10.825 - METERREAD.SERIALNO = 12345
2007-05-24 13:29:10.825 - METERREAD.UNINUMBER = 54321
2007-05-24 13:29:10.825 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:29:10.836 - Parse Script  JSCRIPT
2007-05-24 13:29:10.836 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:29:10.836 - METERREAD.STOPREADING = 1500
2007-05-24 13:29:10.836 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:29:10.846 - METERREAD.METERID = BXT00_GS2
2007-05-24 13:29:10.846 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:29:10.846 - METERREAD.SERIALNO = 12345
2007-05-24 13:29:10.846 - METERREAD.UNINUMBER = 54321
2007-05-24 13:29:10.856 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:29:10.856 - Parse Script  JSCRIPT
2007-05-24 13:29:10.856 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:29:10.856 - METERREAD.STOPREADING = 1500
2007-05-24 13:29:10.856 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:29:17.926 - The rule returned successfully.
2007-05-24 13:29:17.926 - Output created at C:\Training\Output\out_19763.XML
2007-05-24 13:29:17.936 - RT_FilePortal.ProcessFile Finished
=====
```

```
2007-05-24 13:24:21.620 - Performing data conversion...
```


- Data Conversion Saved to message (if applicable)
2007-05-24 13:24:21.620 - Data conversion saved to
C:\LODESTAR\LTMH\Runtime\log\adapter_51136.xml
- Beginning Input Conversion message (if applicable)
2007-05-24 13:24:21.620 - Beginning input conversion.
- Applying Business Rule message (if applicable)
2007-05-24 13:24:21.850 - Applying Business Rule.
- Streaming message (based on xmlstream Business Rule property)
2007-05-24 13:24:22.851 - STREAMING is OFF
- Mapped values, one for each column mapped by the RDL (if applicable - RDL processing only)*
2007-05-24 13:24:22.861 - METERREAD.METERID = BXT00_RES1
- Script Parsing message (if applicable)*
2007-05-24 13:24:22.871 - Parse Script JSCRIPT
- Global Function message (if applicable)*
2007-05-24 13:24:22.871 - BILLDET('KWH')
- Begin Validation Group message (if applicable)*
2007-05-24 13:24:22.871 - ***** Begin Validation Group ***** Meter
- Begin Validation message (if applicable)*
2007-05-24 13:24:22.871 - ***** Begin Validation ***** MeterID
- Validate from message (if applicable - displays type of validation)*
2007-05-24 13:24:22.871 - ***** Validate from SQL *****
- Validation script (if applicable - displays validation script)*
2007-05-24 13:24:22.871 - select meterid from meter where meterid =
'BXT00_RES1'
- Validation Passed/Failed message (if applicable)*
2007-05-24 13:24:22.881 - ***** Validation Passed ***** MeterID
- Validation Group Passed/Failed message (if applicable)*
2007-05-24 13:24:22.881 - ***** Validation Group Passed ***** Meter
- Error XML string (if an error occurred)*
- Success/Failure message*
2007-05-24 13:24:31.374 - The rule returned successfully.
- Business Exception/Rollback message (if an error occurred)*
- Exception message (if an error occurred - contains details of the Java class where the exception occurred)*
- Error message (if an error occurred)*
- Output creation message (if applicable)
2007-05-24 13:24:31.404 - Output created at
C:\Training\Output\out_51137.XML

- Output Conversion Complete message (if applicable)
2007-05-24 13:24:31.404 - Output conversion complete.
- Process Finished message
2007-05-24 13:24:31.414 - RT_FilePortal.ProcessFile Finished
=====

*These lines are repeated for each record within the payload being processed.

Sample Log File - Debug Level 3

```

2007-05-24 13:24:09.212 - Import_CSV_Demo started with process id 1580.
2007-05-24 13:24:09.392 - Initializing process: Import_CSV_Demo
2007-05-24 13:24:09.983 -
=====
=
==                               PROPERTIES                               ==
=====
=
== INPUT_DC_0                      =                                         ==
==   com.lodestarcorp.core.xml.parsers.ColumnCountCSVParser                ==
== INPUT_DC_1                      =                                         ==
==   com.lodestarcorp.portal.data.XSLConverter,C:\LODESTAR\LTMH\CSV_METE    ==
==   R_READS.xml                                                            ==
== INPUT_DC_2                      =                                         ==
==   com.lodestarcorp.portal.data.SchemaValidator,C:\LODESTAR\LTMH\Impor   ==
==   t_CSV_Schema.xsd                                                      ==
== OUTPUT_FILE_DIR                 = C:\Training\Output                    ==
== OUTPUT_FILE_SUFFIX              = .XML                                ==
== OUTPUT_MECHANISM                 = F                                  ==
== POLL_DIRECTORY                  = C:\Training\CSV                      ==
== POLL_EXTENSION                  = txt                                ==
== POLL_INTERVAL                   = 10                                 ==
== RDL_MEMORY_SIZE                 = 3                                  ==
== RULE_NAME                       = Import_CSV_Demo                    ==
== STORAGE_LEVEL                   = 2                                  ==
== WQ_ASSIGNED_TO_USERID           = lou_p                             ==
== WQ_TYPE                         = ADAPTER                            ==
=====
=
2007-05-24 13:24:10.083 - Created data conversion object INPUT_DC_0:
com.lodestarcorp.core.xml.parsers.ColumnCountCSVParser
2007-05-24 13:24:10.363 - Created data conversion object INPUT_DC_1:
com.lodestarcorp.portal.data.XSLConverter
2007-05-24 13:24:10.504 - Created data conversion object INPUT_DC_2:
com.lodestarcorp.portal.data.SchemaValidator
2007-05-24 13:24:10.874 - Business Rule initialization completed.
2007-05-24 13:24:10.904 - Created FileOutputter.
2007-05-24 13:24:10.924 - Initialization completed.
2007-05-24 13:24:21.099 - RT_FilePortal.ProcessFile Started
=====
2007-05-24 13:24:21.620 - Performing data conversion...
2007-05-24 13:24:21.620 - Data conversion saved to
C:\LODESTAR\LTMH\Runtime\log\adapter_51136.xml
2007-05-24 13:24:21.620 - Beginning input conversion.
2007-05-24 13:24:21.850 - Applying Business Rule.
2007-05-24 13:24:22.851 - STREAMING is OFF
2007-05-24 13:24:22.861 - METERREAD.METERID = BXT00_RES1
2007-05-24 13:24:22.861 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:24:22.861 - METERREAD.SERIALNO = 12345
2007-05-24 13:24:22.861 - METERREAD.UNINUMBER = 54321
2007-05-24 13:24:22.861 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:24:22.871 - Parse Script JSCRIPT
2007-05-24 13:24:22.871 - BILLDET('KWH')
2007-05-24 13:24:22.871 - Parse Script JSCRIPT
2007-05-24 13:24:22.871 - METERREAD.BILLDETERMCODE = 1

```

```
2007-05-24 13:24:22.871 - METERREAD.STOPREADING = 1500
2007-05-24 13:24:22.871 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:24:22.871 - ***** Begin Validation Group ***** Meter
2007-05-24 13:24:22.871 - ***** Begin Validation ***** MeterID
2007-05-24 13:24:22.871 - ***** Validate from SQL *****
2007-05-24 13:24:22.871 - select meterid from meter where meterid =
'BXT00_RES1'
2007-05-24 13:24:22.881 - ***** Validation Passed ***** MeterID
2007-05-24 13:24:22.881 - ***** Validation Group Passed ***** Meter
2007-05-24 13:24:22.881 - METERREAD.METERID = BXT00_RES2
2007-05-24 13:24:22.881 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:24:22.891 - METERREAD.SERIALNO = 12345
2007-05-24 13:24:22.891 - METERREAD.UNINUMBER = 54321
2007-05-24 13:24:22.891 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:24:22.891 - Parse Script JSCRIPT
2007-05-24 13:24:22.891 - BILLDET('KWH')
2007-05-24 13:24:22.891 - Parse Script JSCRIPT
2007-05-24 13:24:22.911 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:24:22.911 - METERREAD.STOPREADING = 1500
2007-05-24 13:24:22.911 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:24:22.922 - ***** Begin Validation Group ***** Meter
2007-05-24 13:24:22.992 - ***** Begin Validation ***** MeterID
2007-05-24 13:24:22.992 - ***** Validate from SQL *****
2007-05-24 13:24:22.992 - select meterid from meter where meterid =
'BXT00_RES2'
2007-05-24 13:24:23.002 - ***** Validation Passed ***** MeterID
2007-05-24 13:24:23.052 - ***** Validation Group Passed ***** Meter
2007-05-24 13:24:23.062 - METERREAD.METERID = BXT00_RES3
2007-05-24 13:24:23.062 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:24:23.072 - METERREAD.SERIALNO = 12345
2007-05-24 13:24:23.072 - METERREAD.UNINUMBER = 54321
2007-05-24 13:24:23.072 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:24:23.072 - Parse Script JSCRIPT
2007-05-24 13:24:23.072 - BILLDET('KWH')
2007-05-24 13:24:23.072 - Parse Script JSCRIPT
2007-05-24 13:24:23.072 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:24:23.072 - METERREAD.STOPREADING = 1500
2007-05-24 13:24:23.072 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:24:23.082 - ***** Begin Validation Group ***** Meter
2007-05-24 13:24:23.082 - ***** Begin Validation ***** MeterID
2007-05-24 13:24:23.082 - ***** Validate from SQL *****
2007-05-24 13:24:23.082 - select meterid from meter where meterid =
'BXT00_RES3'
2007-05-24 13:24:23.092 - ***** Validation Passed ***** MeterID
2007-05-24 13:24:23.092 - ***** Validation Group Passed ***** Meter
2007-05-24 13:24:23.102 - METERREAD.METERID = BXT00_RES4
2007-05-24 13:24:23.102 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:24:23.102 - METERREAD.SERIALNO = 12345
2007-05-24 13:24:23.102 - METERREAD.UNINUMBER = 54321
2007-05-24 13:24:23.102 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:24:23.102 - Parse Script JSCRIPT
2007-05-24 13:24:23.112 - BILLDET('KWH')
2007-05-24 13:24:23.112 - Parse Script JSCRIPT
2007-05-24 13:24:23.112 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:24:23.112 - METERREAD.STOPREADING = 1500
2007-05-24 13:24:23.112 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:24:23.112 - ***** Begin Validation Group ***** Meter
2007-05-24 13:24:23.112 - ***** Begin Validation ***** MeterID
2007-05-24 13:24:23.112 - ***** Validate from SQL *****
2007-05-24 13:24:23.112 - select meterid from meter where meterid =
'BXT00_RES4'
2007-05-24 13:24:23.122 - ***** Validation Passed ***** MeterID
2007-05-24 13:24:23.122 - ***** Validation Group Passed ***** Meter
2007-05-24 13:24:23.122 - METERREAD.METERID = BXT00_RES5
2007-05-24 13:24:23.122 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:24:23.122 - METERREAD.SERIALNO = 12345
2007-05-24 13:24:23.122 - METERREAD.UNINUMBER = 54321
2007-05-24 13:24:23.122 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
```

```
2007-05-24 13:24:23.122 - Parse Script  JSCRIPT
2007-05-24 13:24:23.122 - BILLDET('KWH')
2007-05-24 13:24:23.122 - Parse Script  JSCRIPT
2007-05-24 13:24:23.122 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:24:23.132 - METERREAD.STOPREADING = 1500
2007-05-24 13:24:23.132 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:24:23.132 - ***** Begin Validation Group ***** Meter
2007-05-24 13:24:23.132 - ***** Begin Validation ***** MeterID
2007-05-24 13:24:23.132 - ***** Validate from SQL *****
2007-05-24 13:24:23.132 - select meterid from meter where meterid =
'BXT00_RES5'
2007-05-24 13:24:23.132 - ***** Validation Passed ***** MeterID
2007-05-24 13:24:23.132 - ***** Validation Group Passed ***** Meter
2007-05-24 13:24:23.142 - METERREAD.METERID = BXT00_GS1
2007-05-24 13:24:23.142 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:24:23.142 - METERREAD.SERIALNO = 12345
2007-05-24 13:24:23.142 - METERREAD.UNINUMBER = 54321
2007-05-24 13:24:23.142 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:24:23.142 - Parse Script  JSCRIPT
2007-05-24 13:24:23.142 - BILLDET('KWH')
2007-05-24 13:24:23.142 - Parse Script  JSCRIPT
2007-05-24 13:24:23.182 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:24:23.182 - METERREAD.STOPREADING = 1500
2007-05-24 13:24:23.182 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:24:23.192 - ***** Begin Validation Group ***** Meter
2007-05-24 13:24:23.192 - ***** Begin Validation ***** MeterID
2007-05-24 13:24:23.202 - ***** Validate from SQL *****
2007-05-24 13:24:23.202 - select meterid from meter where meterid = 'BXT00_GS1'
2007-05-24 13:24:23.202 - ***** Validation Passed ***** MeterID
2007-05-24 13:24:23.212 - ***** Validation Group Passed ***** Meter
2007-05-24 13:24:23.212 - METERREAD.METERID = BXT00_GS2
2007-05-24 13:24:23.222 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:24:23.222 - METERREAD.SERIALNO = 12345
2007-05-24 13:24:23.222 - METERREAD.UNINUMBER = 54321
2007-05-24 13:24:23.222 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:24:23.222 - Parse Script  JSCRIPT
2007-05-24 13:24:23.222 - BILLDET('KWH')
2007-05-24 13:24:23.222 - Parse Script  JSCRIPT
2007-05-24 13:24:23.222 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:24:23.232 - METERREAD.STOPREADING = 1500
2007-05-24 13:24:23.232 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:24:23.232 - ***** Begin Validation Group ***** Meter
2007-05-24 13:24:23.232 - ***** Begin Validation ***** MeterID
2007-05-24 13:24:23.242 - ***** Validate from SQL *****
2007-05-24 13:24:23.242 - select meterid from meter where meterid = 'BXT00_GS2'
2007-05-24 13:24:23.252 - ***** Validation Passed ***** MeterID
2007-05-24 13:24:23.252 - ***** Validation Group Passed ***** Meter
2007-05-24 13:24:31.374 - The rule returned successfully.
2007-05-24 13:24:31.404 - Output created at C:\Training\Output\out_51137.XML
2007-05-24 13:24:31.404 - Output conversion complete.
2007-05-24 13:24:31.414 - RT_FilePortal.ProcessFile Finished
=====
```

Debug Level 4

At Debug Level 4, the Adapter log file contains the following:

- Service Starting message (including process ID)
2007-05-24 13:23:07.824 - Import_CSV_Demo started with process id 1580.
 - Initializing Process message
2007-05-24 13:23:07.994 - Initializing process: Import_CSV_Demo
 - Runtime Service Properties
2007-05-24 13:23:08.585 -
=====
- ```

=====
==
== PROPERTIES ==
=====
=====
== INPUT_DC_0 = ==
== com.lodestarcorp.core.xml.parsers.ColumnCountCSVParser
==
== INPUT_DC_1 = ==
== com.lodestarcorp.portal.data.XSLConverter,C:\LODESTAR\LTMH\CSV_METE
==
== R_READS.xsl ==
== INPUT_DC_2 = ==
== com.lodestarcorp.portal.data.SchemaValidator,C:\LODESTAR\LTMH\Impor
==
== t_CSV_Schema.xsd ==
== OUTPUT_FILE_DIR = C:\Training\Output ==
== OUTPUT_FILE_SUFFIX = .XML ==
== OUTPUT_MECHANISM = F ==
== POLL_DIRECTORY = C:\Training\CSV ==
== POLL_EXTENSION = txt ==
== POLL_INTERVAL = 10 ==
== RDL_MEMORY_SIZE = 3 ==
== RULE_NAME = Import_CSV_Demo ==
== STORAGE_LEVEL = 2 ==
== WQ_ASSIGNED_TO_USERID = lou_p ==
== WQ_TYPE = ADAPTER ==
=====
=====

```
- Data Conversion Object creation message(s) (if applicable)  
2007-05-24 13:23:08.695 - Created data conversion object INPUT\_DC\_0:  
com.lodestarcorp.core.xml.parsers.ColumnCountCSVParser
  - Business Rule Initialization Complete message  
2007-05-24 13:23:09.486 - Business Rule initialization completed.
  - Output Mechanism message (if applicable)  
2007-05-24 13:23:09.546 - Created FileOutputter.
  - Initialization Complete message  
2007-05-24 13:23:09.546 - Initialization completed.
  - Process Started message  
2007-05-24 13:23:29.715 - RT\_FilePortal.ProcessFile Started  
=====
  - Performing Data Conversion message (if applicable)  
2007-05-24 13:23:30.045 - Performing data conversion...

- Data Conversion Saved to message (if applicable)  
2007-05-24 13:23:30.045 - Data conversion saved to  
C:\LODESTAR\LTMH\Runtime\log\adapter\_1751.xml
- Beginning Input Conversion message (if applicable)  
2007-05-24 13:23:30.055 - Beginning input conversion.
- Applying Business Rule message (if applicable)  
2007-05-24 13:23:30.376 - Applying Business Rule.
- Add Script Function message (if applicable)\*\*  
2007-05-24 13:23:31.457 - \*\*\*\*\* Add Script Function \*\*\*\*\*
- Function message (if applicable)\*\*  
2007-05-24 13:23:31.468 - function BILLDET(sBILLDET) {  
    var retval;  
    switch (sBILLDET) {  
case 'KWH':  
        retval='1';  
        break;  
    }  
    return retval;  
}
- Add Script Function Complete message (if applicable)\*\*  
2007-05-24 13:23:31.457 - \*\*\*\*\* Add Script Function Completed \*\*\*\*\*
- Streaming message (based on xmlstream Business Rule property)  
2007-05-24 13:23:31.488 - STREAMING is OFF
- Start Document Event message (if applicable)  
2007-05-24 13:23:31.488 - \*\*\*\*\* Start Document Event \*\*\*\*\*
- Field To XML string (if applicable - defines RDL mappings)\*\*\*  
2007-05-24 13:23:31.488 - <Field to="METERID" tag="/METERREADS/METERREAD/  
METERID" fmt="" inst="" stdout="" attr="" type="Direct"><![CDATA[]]></  
Field>
- Mapped values, one for each column mapped by the RDL (if applicable - RDL processing  
only)\*  
2007-05-24 13:23:31.608 - METERREAD.METERID = BXT00\_RES1
- Script Parsing message (if applicable)\*  
2007-05-24 13:23:31.608 - Parse Script JSCRIPT
- Global Function message (if applicable)\*  
2007-05-24 13:23:31.608 - BILLDET('KWH')
- Perform Validations message (if applicable)  
2007-05-24 13:23:31.608 - \*\*\*\*\* Perform Validations \*\*\*\*\* METERREAD
- Begin Validation Group message (if applicable)\*  
2007-05-24 13:23:31.608 - \*\*\*\*\* Begin Validation Group \*\*\*\*\* Meter
- Begin Validation message (if applicable)\*  
2007-05-24 13:23:31.618 - \*\*\*\*\* Begin Validation \*\*\*\*\* MeterID

- Validate from message (if applicable - displays type of validation)\*  
2007-05-24 13:23:31.618 - \*\*\*\*\* Validate from SQL \*\*\*\*\*
- Validation script (if applicable - displays validation script)\*  
2007-05-24 13:23:31.618 - select meterid from meter where meterid = 'BXT00\_RES1'
- Validation Passed/Failed message (if applicable)\*  
2007-05-24 13:23:31.708 - \*\*\*\*\* Validation Passed \*\*\*\*\* MeterID
- Validation Group Passed/Failed message (if applicable)\*  
2007-05-24 13:23:31.708 - \*\*\*\*\* Validation Group Passed \*\*\*\*\* Meter
- Perform Table Action message (if applicable)\*  
2007-05-24 13:23:31.708 - \*\*\*\*\* Perform Table Actions \*\*\*\*\*  
METERREAD
- ROW ACTION XML string (if applicable - contains column values and any errors for each record)\*  
2007-05-24 13:23:31.708 - <ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID V="BXT00\_RES1" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /><BILDETERMCODE V="1" /><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></ROW>
- End Document Event message (if applicable)  
2007-05-24 13:23:32.028 - \*\*\*\*\* End Document Event \*\*\*\*\*
- ADAPTER XML string (if applicable - contains column values for entire payload)  
2007-05-24 13:23:32.038 - <ADAPTER><LOCALMEMORY><CUT><INTS></CUT><BABATCH><ADDUPDATE><ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID V="BXT00\_RES1" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /><BILDETERMCODE V="1" /><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></ROW><ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID V="BXT00\_RES2" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /><BILDETERMCODE V="1" /><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></ROW><ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID V="BXT00\_RES3" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /><BILDETERMCODE V="1" /><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></ROW><ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID V="BXT00\_RES4" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /><BILDETERMCODE V="1" /><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></ROW><ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID V="BXT00\_RES5" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /><BILDETERMCODE V="1" /><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></ROW><ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID V="BXT00\_GS1" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /><BILDETERMCODE V="1" /><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></ROW><ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID V="BXT00\_GS2" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /><BILDETERMCODE V="1" /><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></ROW></ADDUPDATE></BABATCH></LOCALMEMORY></ADAPTER>
- Error XML string (if an error occurred)
- Success/Failure message\*  
2007-05-24 13:23:40.420 - The rule returned successfully.

- Business Exception/Rollback message (if an error occurred)\*
- Exception message (if an error occurred - contains details of the Java class where the exception occurred)\*
- Error message (if an error occurred)\*
- Output creation message (if applicable)  
2007-05-24 13:23:40.521 - Output created at C:\Training\Output\out\_1752.XML
- Output Conversion Complete message (if applicable)  
2007-05-24 13:23:40.521 - Output conversion complete.
- Process Finished message  
2007-05-24 13:23:40.541 - RT\_FilePortal.ProcessFile Finished  
=====

\*These lines are repeated for each record within the payload being processed.

\*\* These lines are repeated for each Global Function defined, regardless of whether or not the function is used in the RDL

\*\*\* Repeated for each column mapping defined in the RDL

### Sample Log File - Debug Level 4

```

2007-05-24 13:23:07.824 - Import_CSV_Demo started with process id 1580.
2007-05-24 13:23:07.994 - Initializing process: Import_CSV_Demo
2007-05-24 13:23:08.585 -
=====
=
== PROPERTIES ==
=====
=
== INPUT_DC_0 = ==
== com.lodestarcorp.core.xml.parsers.ColumnCountCSVParser ==
== INPUT_DC_1 = ==
== com.lodestarcorp.portal.data.XSLConverter,C:\LODESTAR\LTMH\CSV_METE ==
== R_READS.xml ==
== INPUT_DC_2 = ==
== com.lodestarcorp.portal.data.SchemaValidator,C:\LODESTAR\LTMH\Impor ==
== t_CSV_Schema.xsd ==
== OUTPUT_FILE_DIR = C:\Training\Output ==
== OUTPUT_FILE_SUFFIX = .XML ==
== OUTPUT_MECHANISM = F ==
== POLL_DIRECTORY = C:\Training\CSV ==
== POLL_EXTENSION = txt ==
== POLL_INTERVAL = 10 ==
== RDL_MEMORY_SIZE = 3 ==
== RULE_NAME = Import_CSV_Demo ==
== STORAGE_LEVEL = 2 ==
== WQ_ASSIGNED_TO_USERID = lou_p ==
== WQ_TYPE = ADAPTER ==
=====
=
2007-05-24 13:23:08.695 - Created data conversion object INPUT_DC_0:
com.lodestarcorp.core.xml.parsers.ColumnCountCSVParser
2007-05-24 13:23:08.975 - Created data conversion object INPUT_DC_1:
com.lodestarcorp.portal.data.XSLConverter
2007-05-24 13:23:09.115 - Created data conversion object INPUT_DC_2:
com.lodestarcorp.portal.data.SchemaValidator
2007-05-24 13:23:09.486 - Business Rule initialization completed.
2007-05-24 13:23:09.546 - Created FileOutputter.
2007-05-24 13:23:09.546 - Initialization completed.
2007-05-24 13:23:29.715 - RT_FilePortal.ProcessFile Started
=====

```



```
2007-05-24 13:23:30.045 - Performing data conversion...
2007-05-24 13:23:30.045 - Data conversion saved to
C:\LODESTAR\LTMH\Runtime\log\adapter_1751.xml
2007-05-24 13:23:30.055 - Beginning input conversion.
2007-05-24 13:23:30.376 - Applying Business Rule.
2007-05-24 13:23:31.457 - ***** Add Script Function *****
2007-05-24 13:23:31.457 - function MeterId(sMeter){
sMeter = sMeter + "-MTR";
return sMeter;}
2007-05-24 13:23:31.457 - ***** Add Script Function Completed *****
2007-05-24 13:23:31.457 - ***** Add Script Function *****
2007-05-24 13:23:31.457 - function UOM(sUOM) {
var retval;
switch (sUOM) {
case 'KWH':
retval='01';
break;
case 'KW':
retval='02';
break;
}
return retval;
}
2007-05-24 13:23:31.457 - ***** Add Script Function Completed *****
2007-05-24 13:23:31.468 - ***** Add Script Function *****
2007-05-24 13:23:31.468 - function BILLDET(sBILLDET) {
var retval;
switch (sBILLDET) {
case 'KWH':
retval='1';
break;
}
return retval;
}
2007-05-24 13:23:31.468 - ***** Add Script Function Completed *****
2007-05-24 13:23:31.468 - ***** Add Script Function *****
2007-05-24 13:23:31.468 - function CreateGUID() {
var x = new ActiveXObject("Scriptlet.TypeLib");
return x.GUID.substring(1, 37);
}
2007-05-24 13:23:31.468 - ***** Add Script Function Completed *****
2007-05-24 13:23:31.488 - STREAMING is OFF
2007-05-24 13:23:31.488 - ***** Start Document Event *****
2007-05-24 13:23:31.488 - <Field to="METERID" tag="/METERREADS/METERREAD/
METERID" fmt="" inst="" stdout="" attr="" type="Direct"><![CDATA[]]></Field>
2007-05-24 13:23:31.488 - <Field to="MANUFACTURER" tag="/METERREADS/METERREAD/
MANUFACTURER" fmt="" inst="" stdout="" attr="" type="Direct"><![CDATA[]]></
Field>
2007-05-24 13:23:31.488 - <Field to="SERIALNO" tag="/METERREADS/METERREAD/
SERIALNO" fmt="" inst="" stdout="" attr="" type="Direct"><![CDATA[]]></Field>
2007-05-24 13:23:31.528 - <Field to="UNINUMBER" tag="/METERREADS/METERREAD/
UNINUMBER" fmt="" inst="" stdout="" attr="" type="Direct"><![CDATA[]]></Field>
2007-05-24 13:23:31.528 - <Field to="METERREADTIME" tag="/METERREADS/METERREAD/
METERREADTIME" fmt="" inst="" stdout="" attr="" type="Direct"><![CDATA[]]></
Field>
2007-05-24 13:23:31.528 - <Field to="BILLDETERMCODE" tag="/METERREADS/
METERREAD/BILLDET" fmt="" inst="BILLDET" stdout="" attr=""
type="PJScript"><![CDATA['%%THIS%%']]></Field>
2007-05-24 13:23:31.608 - <Field to="METERREADMONTH" tag="/METERREADS/
METERREAD/READMONTH" fmt="" inst="" stdout="" attr=""
type="Direct"><![CDATA[]]></Field>
2007-05-24 13:23:31.608 - <Field to="STOPREADING" tag="/METERREADS/METERREAD/
STOPREADING" fmt="" inst="" stdout="" attr="" type="Direct"><![CDATA[]]></
Field>
2007-05-24 13:23:31.608 - METERREAD.METERID = BXT00_RES1
2007-05-24 13:23:31.608 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:23:31.608 - METERREAD.SERIALNO = 12345
2007-05-24 13:23:31.608 - METERREAD.UNINUMBER = 54321
```

```
2007-05-24 13:23:31.608 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:23:31.608 - Parse Script JSCRIPT
2007-05-24 13:23:31.608 - BILLDET('KWH')
2007-05-24 13:23:31.608 - Run Script BILLDET('KWH')
2007-05-24 13:23:31.608 - BILLDET('KWH')
2007-05-24 13:23:31.608 - Parse Script JSCRIPT
2007-05-24 13:23:31.608 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:23:31.608 - METERREAD.STOPREADING = 1500
2007-05-24 13:23:31.608 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:23:31.608 - ***** Perform Validations ***** METERREAD
2007-05-24 13:23:31.608 - ***** Begin Validation Group ***** Meter
2007-05-24 13:23:31.618 - ***** Begin Validation ***** MeterID
2007-05-24 13:23:31.618 - ***** Validate from SQL *****
2007-05-24 13:23:31.618 - select meterid from meter where meterid =
'BXT00_RES1'
2007-05-24 13:23:31.708 - ***** Validation Passed ***** MeterID
2007-05-24 13:23:31.708 - ***** Validation Group Passed ***** Meter
2007-05-24 13:23:31.708 - ***** Perform Table Actions ***** METERREAD
2007-05-24 13:23:31.708 - <ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID
V="BXT00_RES1" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER
V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /><BILLDETERMCODE V="1" /
><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></ROW>
2007-05-24 13:23:31.708 - METERREAD.METERID = BXT00_RES2
2007-05-24 13:23:31.718 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:23:31.718 - METERREAD.SERIALNO = 12345
2007-05-24 13:23:31.718 - METERREAD.UNINUMBER = 54321
2007-05-24 13:23:31.718 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:23:31.718 - Parse Script JSCRIPT
2007-05-24 13:23:31.718 - BILLDET('KWH')
2007-05-24 13:23:31.718 - Run Script BILLDET('KWH')
2007-05-24 13:23:31.718 - BILLDET('KWH')
2007-05-24 13:23:31.718 - Parse Script JSCRIPT
2007-05-24 13:23:31.718 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:23:31.718 - METERREAD.STOPREADING = 1500
2007-05-24 13:23:31.718 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:23:31.718 - ***** Perform Validations ***** METERREAD
2007-05-24 13:23:31.718 - ***** Begin Validation Group ***** Meter
2007-05-24 13:23:31.718 - ***** Begin Validation ***** MeterID
2007-05-24 13:23:31.728 - ***** Validate from SQL *****
2007-05-24 13:23:31.728 - select meterid from meter where meterid =
'BXT00_RES2'
2007-05-24 13:23:31.728 - ***** Validation Passed ***** MeterID
2007-05-24 13:23:31.738 - ***** Validation Group Passed ***** Meter
2007-05-24 13:23:31.738 - ***** Perform Table Actions ***** METERREAD
2007-05-24 13:23:31.738 - <ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID
V="BXT00_RES2" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER
V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /><BILLDETERMCODE V="1" /
><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></ROW>
2007-05-24 13:23:31.798 - METERREAD.METERID = BXT00_RES3
2007-05-24 13:23:31.798 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:23:31.808 - METERREAD.SERIALNO = 12345
2007-05-24 13:23:31.808 - METERREAD.UNINUMBER = 54321
2007-05-24 13:23:31.808 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:23:31.808 - Parse Script JSCRIPT
2007-05-24 13:23:31.808 - BILLDET('KWH')
2007-05-24 13:23:31.808 - Run Script BILLDET('KWH')
2007-05-24 13:23:31.818 - BILLDET('KWH')
2007-05-24 13:23:31.818 - Parse Script JSCRIPT
2007-05-24 13:23:31.818 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:23:31.818 - METERREAD.STOPREADING = 1500
2007-05-24 13:23:31.818 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:23:31.818 - ***** Perform Validations ***** METERREAD
2007-05-24 13:23:31.828 - ***** Begin Validation Group ***** Meter
2007-05-24 13:23:31.828 - ***** Begin Validation ***** MeterID
2007-05-24 13:23:31.828 - ***** Validate from SQL *****
2007-05-24 13:23:31.828 - select meterid from meter where meterid =
'BXT00_RES3'
2007-05-24 13:23:31.838 - ***** Validation Passed ***** MeterID
```

```
2007-05-24 13:23:31.848 - ***** Validation Group Passed ***** Meter
2007-05-24 13:23:31.848 - ***** Perform Table Actions ***** METERREAD
2007-05-24 13:23:31.848 - <ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID
V="BXT00_RES3" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER
V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /><BILLDETERMCODE V="1" /
><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></ROW>
2007-05-24 13:23:31.858 - METERREAD.METERID = BXT00_RES4
2007-05-24 13:23:31.858 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:23:31.858 - METERREAD.SERIALNO = 12345
2007-05-24 13:23:31.858 - METERREAD.UNINUMBER = 54321
2007-05-24 13:23:31.868 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:23:31.868 - Parse Script JSCRIPT
2007-05-24 13:23:31.868 - BILLDET('KWH')
2007-05-24 13:23:31.878 - Run Script BILLDET('KWH')
2007-05-24 13:23:31.878 - BILLDET('KWH')
2007-05-24 13:23:31.878 - Parse Script JSCRIPT
2007-05-24 13:23:31.878 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:23:31.878 - METERREAD.STOPREADING = 1500
2007-05-24 13:23:31.878 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:23:31.878 - ***** Perform Validations ***** METERREAD
2007-05-24 13:23:31.878 - ***** Begin Validation Group ***** Meter
2007-05-24 13:23:31.888 - ***** Begin Validation ***** MeterID
2007-05-24 13:23:31.888 - ***** Validate from SQL *****
2007-05-24 13:23:31.888 - select meterid from meter where meterid =
'BXT00_RES4'
2007-05-24 13:23:31.898 - ***** Validation Passed ***** MeterID
2007-05-24 13:23:31.898 - ***** Validation Group Passed ***** Meter
2007-05-24 13:23:31.898 - ***** Perform Table Actions ***** METERREAD
2007-05-24 13:23:31.898 - <ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID
V="BXT00_RES4" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER
V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /><BILLDETERMCODE V="1" /
><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></ROW>
2007-05-24 13:23:31.898 - METERREAD.METERID = BXT00_RES5
2007-05-24 13:23:31.908 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:23:31.908 - METERREAD.SERIALNO = 12345
2007-05-24 13:23:31.908 - METERREAD.UNINUMBER = 54321
2007-05-24 13:23:31.908 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:23:31.908 - Parse Script JSCRIPT
2007-05-24 13:23:31.908 - BILLDET('KWH')
2007-05-24 13:23:31.908 - Run Script BILLDET('KWH')
2007-05-24 13:23:31.908 - BILLDET('KWH')
2007-05-24 13:23:31.908 - Parse Script JSCRIPT
2007-05-24 13:23:31.908 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:23:31.908 - METERREAD.STOPREADING = 1500
2007-05-24 13:23:31.908 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:23:31.908 - ***** Perform Validations ***** METERREAD
2007-05-24 13:23:31.908 - ***** Begin Validation Group ***** Meter
2007-05-24 13:23:31.908 - ***** Begin Validation ***** MeterID
2007-05-24 13:23:31.908 - ***** Validate from SQL *****
2007-05-24 13:23:31.908 - select meterid from meter where meterid =
'BXT00_RES5'
2007-05-24 13:23:31.918 - ***** Validation Passed ***** MeterID
2007-05-24 13:23:31.918 - ***** Validation Group Passed ***** Meter
2007-05-24 13:23:31.918 - ***** Perform Table Actions ***** METERREAD
2007-05-24 13:23:31.918 - <ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID
V="BXT00_RES5" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER
V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /><BILLDETERMCODE V="1" /
><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></ROW>
2007-05-24 13:23:31.918 - METERREAD.METERID = BXT00_GS1
2007-05-24 13:23:31.918 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:23:31.918 - METERREAD.SERIALNO = 12345
2007-05-24 13:23:31.918 - METERREAD.UNINUMBER = 54321
2007-05-24 13:23:31.918 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:23:31.918 - Parse Script JSCRIPT
2007-05-24 13:23:31.918 - BILLDET('KWH')
2007-05-24 13:23:31.928 - Run Script BILLDET('KWH')
2007-05-24 13:23:31.928 - BILLDET('KWH')
2007-05-24 13:23:31.928 - Parse Script JSCRIPT
```

```
2007-05-24 13:23:31.928 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:23:31.928 - METERREAD.STOPREADING = 1500
2007-05-24 13:23:31.928 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:23:31.928 - ***** Perform Validations ***** METERREAD
2007-05-24 13:23:31.928 - ***** Begin Validation Group ***** Meter
2007-05-24 13:23:31.928 - ***** Begin Validation ***** MeterID
2007-05-24 13:23:31.928 - ***** Validate from SQL *****
2007-05-24 13:23:31.928 - select meterid from meter where meterid = 'BXT00_GS1'
2007-05-24 13:23:31.938 - ***** Validation Passed ***** MeterID
2007-05-24 13:23:31.938 - ***** Validation Group Passed ***** Meter
2007-05-24 13:23:31.938 - ***** Perform Table Actions ***** METERREAD
2007-05-24 13:23:31.948 - <ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID
V="BXT00_GS1" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER
V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /><BILLDETERMCODE V="1" /
><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></ROW>
2007-05-24 13:23:31.998 - METERREAD.METERID = BXT00_GS2
2007-05-24 13:23:31.998 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:23:31.998 - METERREAD.SERIALNO = 12345
2007-05-24 13:23:31.998 - METERREAD.UNINUMBER = 54321
2007-05-24 13:23:31.998 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:23:31.998 - Parse Script JSCRIPT
2007-05-24 13:23:31.998 - BILLDET('KWH')
2007-05-24 13:23:31.998 - Run Script BILLDET('KWH')
2007-05-24 13:23:32.008 - BILLDET('KWH')
2007-05-24 13:23:32.008 - Parse Script JSCRIPT
2007-05-24 13:23:32.008 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:23:32.008 - METERREAD.STOPREADING = 1500
2007-05-24 13:23:32.008 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:23:32.008 - ***** Perform Validations ***** METERREAD
2007-05-24 13:23:32.008 - ***** Begin Validation Group ***** Meter
2007-05-24 13:23:32.018 - ***** Begin Validation ***** MeterID
2007-05-24 13:23:32.018 - ***** Validate from SQL *****
2007-05-24 13:23:32.018 - select meterid from meter where meterid = 'BXT00_GS2'
2007-05-24 13:23:32.028 - ***** Validation Passed ***** MeterID
2007-05-24 13:23:32.028 - ***** Validation Group Passed ***** Meter
2007-05-24 13:23:32.028 - ***** Perform Table Actions ***** METERREAD
2007-05-24 13:23:32.028 - <ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID
V="BXT00_GS2" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER
V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /><BILLDETERMCODE V="1" /
><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></ROW>
2007-05-24 13:23:32.028 - ***** End Document Event *****
2007-05-24 13:23:32.038 - <ADAPTER><LOCALMEMORY><CUT><INTS></>
CUT><BABATCH><ADDUPDATE><ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID
V="BXT00_RES1" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER
V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /><BILLDETERMCODE V="1" /
><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></ROW><ROW
ACTION="ADDUPDATE" TBL="METERREAD"><METERID V="BXT00_RES2" /><MANUFACTURER
V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER V="54321" /><METERREADTIME V="07/
30/2004 00:00:00" /><BILLDETERMCODE V="1" /><STOPREADING V="1500" /
><METERREADMONTH V="07/2004" /></ROW><ROW ACTION="ADDUPDATE"
TBL="METERREAD"><METERID V="BXT00_RES3" /><MANUFACTURER V="METERSRUS" /
><SERIALNO V="12345" /><UNINUMBER V="54321" /><METERREADTIME V="07/30/2004
00:00:00" /><BILLDETERMCODE V="1" /><STOPREADING V="1500" /><METERREADMONTH V="07/
2004" /></ROW><ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID V="BXT00_RES4" /
><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER V="54321" /
><METERREADTIME V="07/30/2004 00:00:00" /><BILLDETERMCODE V="1" /><STOPREADING
V="1500" /><METERREADMONTH V="07/2004" /></ROW><ROW ACTION="ADDUPDATE"
TBL="METERREAD"><METERID V="BXT00_RES5" /><MANUFACTURER V="METERSRUS" /
><SERIALNO V="12345" /><UNINUMBER V="54321" /><METERREADTIME V="07/30/2004
00:00:00" /><BILLDETERMCODE V="1" /><STOPREADING V="1500" /><METERREADMONTH V="07/
2004" /></ROW><ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID V="BXT00_GS1" /
><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER V="54321" /
><METERREADTIME V="07/30/2004 00:00:00" /><BILLDETERMCODE V="1" /><STOPREADING
V="1500" /><METERREADMONTH V="07/2004" /></ROW><ROW ACTION="ADDUPDATE"
TBL="METERREAD"><METERID V="BXT00_GS2" /><MANUFACTURER V="METERSRUS" /><SERIALNO
V="12345" /><UNINUMBER V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /
><BILLDETERMCODE V="1" /><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></
ROW></ADDUPDATE></BABATCH></LOCALMEMORY></ADAPTER>
```

```

2007-05-24 13:23:40.420 - The rule returned successfully.
2007-05-24 13:23:40.521 - Output created at C:\Training\Output\out_1752.XML
2007-05-24 13:23:40.521 - Output conversion complete.
2007-05-24 13:23:40.541 - RT_FilePortal.ProcessFile Finished
=====

```

## Debug Level 5

At Debug Level 5, the Adapter log file contains the following:

- Service Starting message (including process ID)
 

```

2007-05-24 13:21:15.226 - Import_CSV_Demo started with process id 1580.

```
- Initializing Process message
 

```

2007-05-24 13:21:15.416 - Initializing process: Import_CSV_Demo

```
- Runtime Service Properties
 

```

2007-05-24 13:21:16.167 -
=====
====
== PROPERTIES ==
=====
====
== INPUT_DC_0 = ==
== com.lodestarcorp.core.xml.parsers.ColumnCountCSVParser
==
== INPUT_DC_1 = ==
== com.lodestarcorp.portal.data.XSLConverter,C:\LODESTAR\LTMH\CSV_METE
==
== R_READS.xsl
== INPUT_DC_2 = ==
== com.lodestarcorp.portal.data.SchemaValidator,C:\LODESTAR\LTMH\Impor
==
== t_CSV_Schema.xsd
== OUTPUT_FILE_DIR = C:\Training\Output
== OUTPUT_FILE_SUFFIX = .XML
== OUTPUT_MECHANISM = F
== POLL_DIRECTORY = C:\Training\CSV
== POLL_EXTENSION = txt
== POLL_INTERVAL = 10
== RDL_MEMORY_SIZE = 3
== RULE_NAME = Import_CSV_Demo
== STORAGE_LEVEL = 2
== WQ_ASSIGNED_TO_USERID = lou_p
== WQ_TYPE = ADAPTER
=====
====

```
- Data Conversion Object creation message(s) (if applicable)
 

```

2007-05-24 13:21:16.278 - Created data conversion object INPUT_DC_0:
com.lodestarcorp.core.xml.parsers.ColumnCountCSVParser

```
- Business Rule Initialization Complete message
 

```

2007-05-24 13:21:19.242 - Business Rule initialization completed.

```
- Output Mechanism message (if applicable)
 

```

2007-05-24 13:21:19.372 - Created FileOutputter

```
- Initialization Complete message
 

```

2007-05-24 13:21:19.372 - Initialization completed.

```

- Process Started message  

```
2007-05-24 13:21:39.704 - RT_FilePortal.ProcessFile Started
=====
```
- Performing Data Conversion message (if applicable)  

```
2007-05-24 13:21:42.829 - Performing data conversion...
```
- Data Conversion Saved to message (if applicable)  

```
2007-05-24 13:21:42.849 - Data conversion saved to
C:\LODESTAR\LTMH\Runtime\log\adapter_31323.xml
```
- Beginning Input Conversion message (if applicable)  

```
2007-05-24 13:21:42.849 - Beginning input conversion.
```
- Applying Business Rule message (if applicable)  

```
2007-05-24 13:21:43.049 - Applying Business Rule.
```
- Add Script Function message (if applicable)\*\*  

```
2007-05-24 13:21:44.862 - ***** Add Script Function *****
```
- Function message (if applicable)\*\*  

```
2007-05-24 13:21:44.872 - function BILLDET(sBILLDET) {
 var retval;
 switch (sBILLDET) {
case 'KWH':
 retval='1';
 break;
 }
 return retval;
}
```
- Add Script Function Complete message (if applicable)\*\*  

```
2007-05-24 13:21:44.872 - ***** Add Script Function Completed *****
```
- Streaming message (based on xmlstream Business Rule property)  

```
2007-05-24 13:21:47.887 - STREAMING is OFF
```
- Mode message (Batch or Event)  

```
2007-05-24 13:21:47.917 - MODE = BATCH
```
- Debug Level message  

```
2007-05-24 13:21:47.917 - Debug Level = 5
```
- Start Document Event message (if applicable)  

```
2007-05-24 13:21:47.917 - ***** Start Document Event *****
```
- Field To XML string (if applicable - defines RDL mappings)\*\*  

```
2007-05-24 13:21:47.917 - <Field to="METERID" tag="/METERREADS/METERREAD/
METERID" fmt="" inst="" stdout="" attr="" type="Direct"><![CDATA[]]></
Field>
```
- XML Parsing messages (if applicable - displays how XML payload is parsed)\*  

```
2007-05-24 13:21:48.007 - /METERREADS
2007-05-24 13:21:48.007 - /METERREADS/METERREAD
2007-05-24 13:21:48.007 - /METERREADS/METERREAD/METERID
```
- Mapped values, one for each column mapped by the RDL (if applicable - RDL processing only)\*  

```
2007-05-24 13:21:48.007 - METERREAD.METERID = BXT00_RES1
```

- Script Parsing message (if applicable)\*  
2007-05-24 13:21:48.027 - Parse Script JSCRIPT
- Global Function message (if applicable)\*  
2007-05-24 13:21:48.027 - BILLDET('KWH')
- Perform Validations message (if applicable)  
2007-05-24 13:21:48.027 - \*\*\*\*\* Perform Validations \*\*\*\*\* METERREAD
- Begin Validation Group message (if applicable)\*  
2007-05-24 13:21:48.027 - \*\*\*\*\* Begin Validation Group \*\*\*\*\* Meter
- Begin Validation message (if applicable)\*  
2007-05-24 13:21:48.027 - \*\*\*\*\* Begin Validation \*\*\*\*\* MeterID
- Validate from message (if applicable - displays type of validation)\*  
2007-05-24 13:21:48.027 - \*\*\*\*\* Validate from SQL \*\*\*\*\*
- Validation script (if applicable - displays validation script)\*  
2007-05-24 13:21:48.027 - select meterid from meter where meterid =  
'BXT00\_RES1'
- Validation Passed/Failed message (if applicable)\*  
2007-05-24 13:21:48.157 - \*\*\*\*\* Validation Passed \*\*\*\*\* MeterID
- Validation Group Passed/Failed message (if applicable)\*  
2007-05-24 13:21:48.157 - \*\*\*\*\* Validation Group Passed \*\*\*\*\* Meter
- Perform Table Action message (if applicable)\*  
2007-05-24 13:21:48.167 - \*\*\*\*\* Perform Table Actions \*\*\*\*\*  
METERREAD
- ROW ACTION XML string (if applicable - contains column values and any errors for each record)\*  
2007-05-24 13:21:48.167 - <ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID  
V="BXT00\_RES1" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /  
><UNINUMBER V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /  
><BILLDETERMCODE V="1" /><STOPREADING V="1500" /><METERREADMONTH V="07/  
2004" /></ROW>
- End Document Event message (if applicable)  
2007-05-24 13:21:48.758 - \*\*\*\*\* End Document Event \*\*\*\*\*

- ADAPTER XML string (if applicable - contains column values for entire payload)

```
2007-05-24 13:21:48.758 - <ADAPTER><LOCALMEMORY><CUT><INTS></>
CUT><BABATCH><ADDUPDATE><ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID
V="BXT00_RES1"/><MANUFACTURER V="METERSRUS"/><SERIALNO V="12345"/
><UNINUMBER V="54321"/><METERREADTIME V="07/30/2004 00:00:00"/
><BILLDETERMCODE V="1"/><STOPREADING V="1500"/><METERREADMONTH V="07/2004"/
></ROW><ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID V="BXT00_RES2"/
><MANUFACTURER V="METERSRUS"/><SERIALNO V="12345"/><UNINUMBER V="54321"/
><METERREADTIME V="07/30/2004 00:00:00"/><BILLDETERMCODE V="1"/
><STOPREADING V="1500"/><METERREADMONTH V="07/2004"/></ROW><ROW
ACTION="ADDUPDATE" TBL="METERREAD"><METERID V="BXT00_RES3"/><MANUFACTURER
V="METERSRUS"/><SERIALNO V="12345"/><UNINUMBER V="54321"/><METERREADTIME
V="07/30/2004 00:00:00"/><BILLDETERMCODE V="1"/><STOPREADING V="1500"/
><METERREADMONTH V="07/2004"/></ROW><ROW ACTION="ADDUPDATE"
TBL="METERREAD"><METERID V="BXT00_RES4"/><MANUFACTURER V="METERSRUS"/
><SERIALNO V="12345"/><UNINUMBER V="54321"/><METERREADTIME V="07/30/2004
00:00:00"/><BILLDETERMCODE V="1"/><STOPREADING V="1500"/><METERREADMONTH
V="07/2004"/></ROW><ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID
V="BXT00_RES5"/><MANUFACTURER V="METERSRUS"/><SERIALNO V="12345"/
><UNINUMBER V="54321"/><METERREADTIME V="07/30/2004 00:00:00"/
><BILLDETERMCODE V="1"/><STOPREADING V="1500"/><METERREADMONTH V="07/2004"/
></ROW><ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID V="BXT00_GS1"/
><MANUFACTURER V="METERSRUS"/><SERIALNO V="12345"/><UNINUMBER V="54321"/
><METERREADTIME V="07/30/2004 00:00:00"/><BILLDETERMCODE V="1"/
><STOPREADING V="1500"/><METERREADMONTH V="07/2004"/></ROW><ROW
ACTION="ADDUPDATE" TBL="METERREAD"><METERID V="BXT00_GS2"/><MANUFACTURER
V="METERSRUS"/><SERIALNO V="12345"/><UNINUMBER V="54321"/><METERREADTIME
V="07/30/2004 00:00:00"/><BILLDETERMCODE V="1"/><STOPREADING V="1500"/
><METERREADMONTH V="07/2004"/></ROW></ADDUPDATE></BABATCH></LOCALMEMORY></>
ADAPTER>
```

- Error XML string (if an error occurred)

- Success/Failure message\*

```
2007-05-24 13:21:58.053 - The rule returned successfully.
```

- Business Exception/Rollback message (if an error occurred)\*

- Exception message (if an error occurred - contains details of the Java class where the exception occurred)\*

- Error message (if an error occurred)\*

- Output creation message (if applicable)

```
2007-05-24 13:21:58.083 - Output created at
C:\Training\Output\out_31324.XML
```

- Output Conversion Complete message (if applicable)

```
2007-05-24 13:21:58.083 - Output conversion complete.
```

- Process Finished message

```
2007-05-24 13:21:58.103 - RT_FilePortal.ProcessFile Finished
=====
```

\*These lines are repeated for each record within the payload being processed.

\*\* These lines are repeated for each Global Function defined, regardless of whether or not the function is used in the RDL

\*\*\* Repeated for each column mapping defined in the RDL



## Sample Log File - Debug Level 5

```

2007-05-24 13:21:15.226 - Import_CSV_Demo started with process id 1580.
2007-05-24 13:21:15.416 - Initializing process: Import_CSV_Demo
2007-05-24 13:21:16.167 -
=====
==
== PROPERTIES ==
=====
=
== INPUT_DC_0 = ==
== com.lodestarcorp.core.xml.parsers.ColumnCountCSVParser ==
== INPUT_DC_1 = ==
== com.lodestarcorp.portal.data.XSLConverter,C:\LODESTAR\LTMH\CSV_METE ==
== R_READS.xsl ==
== INPUT_DC_2 = ==
== com.lodestarcorp.portal.data.SchemaValidator,C:\LODESTAR\LTMH\Impor ==
== t_CSV_Schema.xsd ==
== OUTPUT_FILE_DIR = C:\Training\Output ==
== OUTPUT_FILE_SUFFIX = .XML ==
== OUTPUT_MECHANISM = F ==
== POLL_DIRECTORY = C:\Training\CSV ==
== POLL_EXTENSION = txt ==
== POLL_INTERVAL = 10 ==
== RDL_MEMORY_SIZE = 3 ==
== RULE_NAME = Import_CSV_Demo ==
== STORAGE_LEVEL = 2 ==
== WQ_ASSIGNED_TO_USERID = lou_p ==
== WQ_TYPE = ADAPTER ==
=====
=
2007-05-24 13:21:16.278 - Created data conversion object INPUT_DC_0:
com.lodestarcorp.core.xml.parsers.ColumnCountCSVParser
2007-05-24 13:21:17.590 - Created data conversion object INPUT_DC_1:
com.lodestarcorp.portal.data.XSLConverter
2007-05-24 13:21:17.770 - Created data conversion object INPUT_DC_2:
com.lodestarcorp.portal.data.SchemaValidator
2007-05-24 13:21:19.242 - Business Rule initialization completed.
2007-05-24 13:21:19.372 - Created FileOutputter.
2007-05-24 13:21:19.372 - Initialization completed.
2007-05-24 13:21:39.704 - RT_FilePortal.ProcessFile Started
=====
2007-05-24 13:21:42.829 - Performing data conversion...
2007-05-24 13:21:42.849 - Data conversion saved to
C:\LODESTAR\LTMH\Runtime\log\adapter_31323.xml
2007-05-24 13:21:42.849 - Beginning input conversion.
2007-05-24 13:21:43.049 - Applying Business Rule.
2007-05-24 13:21:44.862 - ***** Add Script Function *****
2007-05-24 13:21:44.862 - function MeterId(sMeter){
sMeter = sMeter + "-MTR";
return sMeter;}
2007-05-24 13:21:44.872 - ***** Add Script Function Completed *****
2007-05-24 13:21:44.872 - ***** Add Script Function *****
2007-05-24 13:21:44.872 - function UOM(sUOM) {
var retVal;
switch (sUOM) {
case 'KWH':
retVal='01';
break;
case 'KW':
retVal='02';
break;
}
return retVal;
}
2007-05-24 13:21:44.872 - ***** Add Script Function Completed *****
2007-05-24 13:21:44.872 - ***** Add Script Function *****

```

```
2007-05-24 13:21:44.872 - function BILLDET(sBILLDET) {
 var retval;
 switch (sBILLDET) {
 case 'KWH':
 retval='1';
 break;
 }
 return retval;
}
2007-05-24 13:21:44.872 - ***** Add Script Function Completed *****
2007-05-24 13:21:44.872 - ***** Add Script Function *****
2007-05-24 13:21:44.872 - function CreateGUID() {
 var x = new ActiveXObject("Scriptlet.TypeLib");
 return x.GUID.substring(1, 37);
}
2007-05-24 13:21:44.872 - ***** Add Script Function Completed *****
2007-05-24 13:21:47.887 - STREAMING is OFF
2007-05-24 13:21:47.917 - MODE = BATCH
2007-05-24 13:21:47.917 - Debug Level = 5
2007-05-24 13:21:47.917 - ***** Start Document Event *****
2007-05-24 13:21:47.917 - <Field to="METERID" tag="/METERREADS/METERREAD/
METERID" fmt="" inst="" stdout="" attr="" type="Direct"><![CDATA[]]></Field>
2007-05-24 13:21:47.917 - <Field to="MANUFACTURER" tag="/METERREADS/METERREAD/
MANUFACTURER" fmt="" inst="" stdout="" attr="" type="Direct"><![CDATA[]]></
Field>
2007-05-24 13:21:47.917 - <Field to="SERIALNO" tag="/METERREADS/METERREAD/
SERIALNO" fmt="" inst="" stdout="" attr="" type="Direct"><![CDATA[]]></Field>
2007-05-24 13:21:47.927 - <Field to="UNINUMBER" tag="/METERREADS/METERREAD/
UNINUMBER" fmt="" inst="" stdout="" attr="" type="Direct"><![CDATA[]]></Field>
2007-05-24 13:21:47.927 - <Field to="METERREADTIME" tag="/METERREADS/METERREAD/
METERREADTIME" fmt="" inst="" stdout="" attr="" type="Direct"><![CDATA[]]></
Field>
2007-05-24 13:21:47.927 - <Field to="BILLDETERMCODE" tag="/METERREADS/
METERREAD/BILLDET" fmt="" inst="BILLDET" stdout="" attr=""
type="PJScript"><![CDATA['%%THIS%']]></Field>
2007-05-24 13:21:48.007 - <Field to="METERREADMONTH" tag="/METERREADS/
METERREAD/READMONTH" fmt="" inst="" stdout="" attr=""
type="Direct"><![CDATA[]]></Field>
2007-05-24 13:21:48.007 - <Field to="STOPREADING" tag="/METERREADS/METERREAD/
STOPREADING" fmt="" inst="" stdout="" attr="" type="Direct"><![CDATA[]]></
Field>
2007-05-24 13:21:48.007 - /METERREADS
2007-05-24 13:21:48.007 - /METERREADS/METERREAD
2007-05-24 13:21:48.007 - /METERREADS/METERREAD/METERID
2007-05-24 13:21:48.007 - METERREAD.METERID = BXT00_RES1
2007-05-24 13:21:48.017 - /METERREADS/METERREAD/MANUFACTURER
2007-05-24 13:21:48.017 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:21:48.017 - /METERREADS/METERREAD/SERIALNO
2007-05-24 13:21:48.017 - METERREAD.SERIALNO = 12345
2007-05-24 13:21:48.017 - /METERREADS/METERREAD/UNINUMBER
2007-05-24 13:21:48.017 - METERREAD.UNINUMBER = 54321
2007-05-24 13:21:48.017 - /METERREADS/METERREAD/METERREADTIME
2007-05-24 13:21:48.017 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:21:48.017 - /METERREADS/METERREAD/BILLDET
2007-05-24 13:21:48.027 - Parse Script JSCRIPT
2007-05-24 13:21:48.027 - BILLDET('KWH')
2007-05-24 13:21:48.027 - Run Script BILLDET('KWH')
2007-05-24 13:21:48.027 - BILLDET('KWH')
2007-05-24 13:21:48.027 - Parse Script JSCRIPT
2007-05-24 13:21:48.027 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:21:48.027 - /METERREADS/METERREAD/STOPREADING
2007-05-24 13:21:48.027 - METERREAD.STOPREADING = 1500
2007-05-24 13:21:48.027 - /METERREADS/METERREAD/READMONTH
2007-05-24 13:21:48.027 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:21:48.027 - ***** Perform Validations ***** METERREAD
2007-05-24 13:21:48.027 - ***** Begin Validation Group ***** Meter
2007-05-24 13:21:48.027 - ***** Begin Validation ***** MeterID
2007-05-24 13:21:48.027 - ***** Validate from SQL *****
```

```
2007-05-24 13:21:48.027 - select meterid from meter where meterid =
'BXT00_RES1'
2007-05-24 13:21:48.157 - ***** Validation Passed ***** MeterID
2007-05-24 13:21:48.157 - ***** Validation Group Passed ***** Meter
2007-05-24 13:21:48.167 - ***** Perform Table Actions ***** METERREAD
2007-05-24 13:21:48.167 - <ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID
V="BXT00_RES1" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER
V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /><BILLDETERMCODE V="1" /
><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></ROW>
2007-05-24 13:21:48.167 - /METERREADS/METERREAD
2007-05-24 13:21:48.167 - /METERREADS/METERREAD/METERID
2007-05-24 13:21:48.167 - METERREAD.METERID = BXT00_RES2
2007-05-24 13:21:48.167 - /METERREADS/METERREAD/MANUFACTURER
2007-05-24 13:21:48.167 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:21:48.167 - /METERREADS/METERREAD/SERIALNO
2007-05-24 13:21:48.167 - METERREAD.SERIALNO = 12345
2007-05-24 13:21:48.167 - /METERREADS/METERREAD/UNINUMBER
2007-05-24 13:21:48.167 - METERREAD.UNINUMBER = 54321
2007-05-24 13:21:48.167 - /METERREADS/METERREAD/METERREADTIME
2007-05-24 13:21:48.167 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:21:48.167 - /METERREADS/METERREAD/BILLDET
2007-05-24 13:21:48.167 - Parse Script JSCRIPT
2007-05-24 13:21:48.167 - BILLDET('KWH')
2007-05-24 13:21:48.167 - Run Script BILLDET('KWH')
2007-05-24 13:21:48.167 - BILLDET('KWH')
2007-05-24 13:21:48.177 - Parse Script JSCRIPT
2007-05-24 13:21:48.177 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:21:48.177 - /METERREADS/METERREAD/STOPREADING
2007-05-24 13:21:48.177 - METERREAD.STOPREADING = 1500
2007-05-24 13:21:48.177 - /METERREADS/METERREAD/READMONTH
2007-05-24 13:21:48.177 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:21:48.177 - ***** Perform Validations ***** METERREAD
2007-05-24 13:21:48.177 - ***** Begin Validation Group ***** Meter
2007-05-24 13:21:48.177 - ***** Begin Validation ***** MeterID
2007-05-24 13:21:48.177 - ***** Validate from SQL *****
2007-05-24 13:21:48.177 - select meterid from meter where meterid =
'BXT00_RES2'
2007-05-24 13:21:48.187 - ***** Validation Passed ***** MeterID
2007-05-24 13:21:48.187 - ***** Validation Group Passed ***** Meter
2007-05-24 13:21:48.197 - ***** Perform Table Actions ***** METERREAD
2007-05-24 13:21:48.207 - <ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID
V="BXT00_RES2" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER
V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /><BILLDETERMCODE V="1" /
><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></ROW>
2007-05-24 13:21:48.237 - /METERREADS/METERREAD
2007-05-24 13:21:48.237 - /METERREADS/METERREAD/METERID
2007-05-24 13:21:48.237 - METERREAD.METERID = BXT00_RES3
2007-05-24 13:21:48.237 - /METERREADS/METERREAD/MANUFACTURER
2007-05-24 13:21:48.237 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:21:48.237 - /METERREADS/METERREAD/SERIALNO
2007-05-24 13:21:48.237 - METERREAD.SERIALNO = 12345
2007-05-24 13:21:48.237 - /METERREADS/METERREAD/UNINUMBER
2007-05-24 13:21:48.247 - METERREAD.UNINUMBER = 54321
2007-05-24 13:21:48.247 - /METERREADS/METERREAD/METERREADTIME
2007-05-24 13:21:48.247 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:21:48.257 - /METERREADS/METERREAD/BILLDET
2007-05-24 13:21:48.257 - Parse Script JSCRIPT
2007-05-24 13:21:48.257 - BILLDET('KWH')
2007-05-24 13:21:48.257 - Run Script BILLDET('KWH')
2007-05-24 13:21:48.257 - BILLDET('KWH')
2007-05-24 13:21:48.257 - Parse Script JSCRIPT
2007-05-24 13:21:48.257 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:21:48.257 - /METERREADS/METERREAD/STOPREADING
2007-05-24 13:21:48.257 - METERREAD.STOPREADING = 1500
2007-05-24 13:21:48.267 - /METERREADS/METERREAD/READMONTH
2007-05-24 13:21:48.267 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:21:48.287 - ***** Perform Validations ***** METERREAD
2007-05-24 13:21:48.287 - ***** Begin Validation Group ***** Meter
```

```
2007-05-24 13:21:48.287 - ***** Begin Validation ***** MeterID
2007-05-24 13:21:48.297 - ***** Validate from SQL *****
2007-05-24 13:21:48.297 - select meterid from meter where meterid =
'BXT00_RES3'
2007-05-24 13:21:48.307 - ***** Validation Passed ***** MeterID
2007-05-24 13:21:48.307 - ***** Validation Group Passed ***** Meter
2007-05-24 13:21:48.307 - ***** Perform Table Actions ***** METERREAD
2007-05-24 13:21:48.317 - <ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID
V="BXT00_RES3" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER
V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /><BILLDETERMCODE V="1" /
><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></ROW>
2007-05-24 13:21:48.317 - /METERREADS/METERREAD
2007-05-24 13:21:48.317 - /METERREADS/METERREAD/METERID
2007-05-24 13:21:48.317 - METERREAD.METERID = BXT00_RES4
2007-05-24 13:21:48.327 - /METERREADS/METERREAD/MANUFACTURER
2007-05-24 13:21:48.327 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:21:48.327 - /METERREADS/METERREAD/SERIALNO
2007-05-24 13:21:48.327 - METERREAD.SERIALNO = 12345
2007-05-24 13:21:48.327 - /METERREADS/METERREAD/UNINUMBER
2007-05-24 13:21:48.327 - METERREAD.UNINUMBER = 54321
2007-05-24 13:21:48.327 - /METERREADS/METERREAD/METERREADTIME
2007-05-24 13:21:48.327 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:21:48.338 - /METERREADS/METERREAD/BILLDET
2007-05-24 13:21:48.338 - Parse Script JSCRIPT
2007-05-24 13:21:48.338 - BILLDET('KWH')
2007-05-24 13:21:48.338 - Run Script BILLDET('KWH')
2007-05-24 13:21:48.338 - BILLDET('KWH')
2007-05-24 13:21:48.338 - Parse Script JSCRIPT
2007-05-24 13:21:48.338 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:21:48.338 - /METERREADS/METERREAD/STOPREADING
2007-05-24 13:21:48.338 - METERREAD.STOPREADING = 1500
2007-05-24 13:21:48.348 - /METERREADS/METERREAD/READMONTH
2007-05-24 13:21:48.348 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:21:48.348 - ***** Perform Validations ***** METERREAD
2007-05-24 13:21:48.358 - ***** Begin Validation Group ***** Meter
2007-05-24 13:21:48.358 - ***** Begin Validation ***** MeterID
2007-05-24 13:21:48.358 - ***** Validate from SQL *****
2007-05-24 13:21:48.358 - select meterid from meter where meterid =
'BXT00_RES4'
2007-05-24 13:21:48.368 - ***** Validation Passed ***** MeterID
2007-05-24 13:21:48.368 - ***** Validation Group Passed ***** Meter
2007-05-24 13:21:48.368 - ***** Perform Table Actions ***** METERREAD
2007-05-24 13:21:48.378 - <ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID
V="BXT00_RES4" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER
V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /><BILLDETERMCODE V="1" /
><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></ROW>
2007-05-24 13:21:48.378 - /METERREADS/METERREAD
2007-05-24 13:21:48.378 - /METERREADS/METERREAD/METERID
2007-05-24 13:21:48.378 - METERREAD.METERID = BXT00_RES5
2007-05-24 13:21:48.378 - /METERREADS/METERREAD/MANUFACTURER
2007-05-24 13:21:48.378 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:21:48.378 - /METERREADS/METERREAD/SERIALNO
2007-05-24 13:21:48.378 - METERREAD.SERIALNO = 12345
2007-05-24 13:21:48.378 - /METERREADS/METERREAD/UNINUMBER
2007-05-24 13:21:48.378 - METERREAD.UNINUMBER = 54321
2007-05-24 13:21:48.378 - /METERREADS/METERREAD/METERREADTIME
2007-05-24 13:21:48.378 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:21:48.388 - /METERREADS/METERREAD/BILLDET
2007-05-24 13:21:48.388 - Parse Script JSCRIPT
2007-05-24 13:21:48.388 - BILLDET('KWH')
2007-05-24 13:21:48.388 - Run Script BILLDET('KWH')
2007-05-24 13:21:48.388 - BILLDET('KWH')
2007-05-24 13:21:48.388 - Parse Script JSCRIPT
2007-05-24 13:21:48.388 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:21:48.388 - /METERREADS/METERREAD/STOPREADING
2007-05-24 13:21:48.388 - METERREAD.STOPREADING = 1500
2007-05-24 13:21:48.388 - /METERREADS/METERREAD/READMONTH
2007-05-24 13:21:48.388 - METERREAD.METERREADMONTH = 07/2004
```

```
2007-05-24 13:21:48.388 - ***** Perform Validations ***** METERREAD
2007-05-24 13:21:48.388 - ***** Begin Validation Group ***** Meter
2007-05-24 13:21:48.388 - ***** Begin Validation ***** MeterID
2007-05-24 13:21:48.388 - ***** Validate from SQL *****
2007-05-24 13:21:48.388 - select meterid from meter where meterid =
'BXT00_RES5'
2007-05-24 13:21:48.398 - ***** Validation Passed ***** MeterID
2007-05-24 13:21:48.398 - ***** Validation Group Passed ***** Meter
2007-05-24 13:21:48.398 - ***** Perform Table Actions ***** METERREAD
2007-05-24 13:21:48.398 - <ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID
V="BXT00_RES5" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER
V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /><BILLDETERMCODE V="1" /
><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></ROW>
2007-05-24 13:21:48.398 - /METERREADS/METERREAD
2007-05-24 13:21:48.398 - /METERREADS/METERREAD/METERID
2007-05-24 13:21:48.398 - METERREAD.METERID = BXT00_GS1
2007-05-24 13:21:48.398 - /METERREADS/METERREAD/MANUFACTURER
2007-05-24 13:21:48.398 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:21:48.398 - /METERREADS/METERREAD/SERIALNO
2007-05-24 13:21:48.398 - METERREAD.SERIALNO = 12345
2007-05-24 13:21:48.408 - /METERREADS/METERREAD/UNINUMBER
2007-05-24 13:21:48.408 - METERREAD.UNINUMBER = 54321
2007-05-24 13:21:48.408 - /METERREADS/METERREAD/METERREADTIME
2007-05-24 13:21:48.408 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:21:48.408 - /METERREADS/METERREAD/BILLDET
2007-05-24 13:21:48.408 - Parse Script JSCRIPT
2007-05-24 13:21:48.408 - BILLDET('KWH')
2007-05-24 13:21:48.408 - Run Script BILLDET('KWH')
2007-05-24 13:21:48.408 - BILLDET('KWH')
2007-05-24 13:21:48.408 - Parse Script JSCRIPT
2007-05-24 13:21:48.408 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:21:48.408 - /METERREADS/METERREAD/STOPREADING
2007-05-24 13:21:48.408 - METERREAD.STOPREADING = 1500
2007-05-24 13:21:48.408 - /METERREADS/METERREAD/READMONTH
2007-05-24 13:21:48.408 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:21:48.408 - ***** Perform Validations ***** METERREAD
2007-05-24 13:21:48.408 - ***** Begin Validation Group ***** Meter
2007-05-24 13:21:48.408 - ***** Begin Validation ***** MeterID
2007-05-24 13:21:48.418 - ***** Validate from SQL *****
2007-05-24 13:21:48.418 - select meterid from meter where meterid = 'BXT00_GS1'
2007-05-24 13:21:48.428 - ***** Validation Passed ***** MeterID
2007-05-24 13:21:48.428 - ***** Validation Group Passed ***** Meter
2007-05-24 13:21:48.428 - ***** Perform Table Actions ***** METERREAD
2007-05-24 13:21:48.438 - <ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID
V="BXT00_GS1" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER
V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /><BILLDETERMCODE V="1" /
><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></ROW>
2007-05-24 13:21:48.578 - /METERREADS/METERREAD
2007-05-24 13:21:48.578 - /METERREADS/METERREAD/METERID
2007-05-24 13:21:48.578 - METERREAD.METERID = BXT00_GS2
2007-05-24 13:21:48.578 - /METERREADS/METERREAD/MANUFACTURER
2007-05-24 13:21:48.578 - METERREAD.MANUFACTURER = METERSRUS
2007-05-24 13:21:48.588 - /METERREADS/METERREAD/SERIALNO
2007-05-24 13:21:48.588 - METERREAD.SERIALNO = 12345
2007-05-24 13:21:48.588 - /METERREADS/METERREAD/UNINUMBER
2007-05-24 13:21:48.588 - METERREAD.UNINUMBER = 54321
2007-05-24 13:21:48.588 - /METERREADS/METERREAD/METERREADTIME
2007-05-24 13:21:48.588 - METERREAD.METERREADTIME = 07/30/2004 00:00:00
2007-05-24 13:21:48.588 - /METERREADS/METERREAD/BILLDET
2007-05-24 13:21:48.588 - Parse Script JSCRIPT
2007-05-24 13:21:48.588 - BILLDET('KWH')
2007-05-24 13:21:48.588 - Run Script BILLDET('KWH')
2007-05-24 13:21:48.588 - BILLDET('KWH')
2007-05-24 13:21:48.598 - Parse Script JSCRIPT
2007-05-24 13:21:48.598 - METERREAD.BILLDETERMCODE = 1
2007-05-24 13:21:48.598 - /METERREADS/METERREAD/STOPREADING
2007-05-24 13:21:48.608 - METERREAD.STOPREADING = 1500
2007-05-24 13:21:48.608 - /METERREADS/METERREAD/READMONTH
```

```
2007-05-24 13:21:48.608 - METERREAD.METERREADMONTH = 07/2004
2007-05-24 13:21:48.658 - ***** Perform Validations ***** METERREAD
2007-05-24 13:21:48.668 - ***** Begin Validation Group ***** Meter
2007-05-24 13:21:48.668 - ***** Begin Validation ***** MeterID
2007-05-24 13:21:48.668 - ***** Validate from SQL *****
2007-05-24 13:21:48.668 - select meterid from meter where meterid = 'BXT00_GS2'
2007-05-24 13:21:48.748 - ***** Validation Passed ***** MeterID
2007-05-24 13:21:48.748 - ***** Validation Group Passed ***** Meter
2007-05-24 13:21:48.748 - ***** Perform Table Actions ***** METERREAD
2007-05-24 13:21:48.758 - <ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID
V="BXT00_GS2" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER
V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /><BILDETERMCODE V="1" /
><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></ROW>
2007-05-24 13:21:48.758 - ***** End Document Event *****
2007-05-24 13:21:48.758 - <ADAPTER><LOCALMEMORY><CUT><INTS></>
CUT><BABATCH><ADDUPDATE><ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID
V="BXT00_RES1" /><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER
V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /><BILDETERMCODE V="1" /
><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></ROW><ROW
ACTION="ADDUPDATE" TBL="METERREAD"><METERID V="BXT00_RES2" /><MANUFACTURER
V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER V="54321" /><METERREADTIME V="07/
30/2004 00:00:00" /><BILDETERMCODE V="1" /><STOPREADING V="1500" /
><METERREADMONTH V="07/2004" /></ROW><ROW ACTION="ADDUPDATE"
TBL="METERREAD"><METERID V="BXT00_RES3" /><MANUFACTURER V="METERSRUS" /
><SERIALNO V="12345" /><UNINUMBER V="54321" /><METERREADTIME V="07/30/2004
00:00:00" /><BILDETERMCODE V="1" /><STOPREADING V="1500" /><METERREADMONTH V="07/
2004" /></ROW><ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID V="BXT00_RES4" /
><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER V="54321" /
><METERREADTIME V="07/30/2004 00:00:00" /><BILDETERMCODE V="1" /><STOPREADING
V="1500" /><METERREADMONTH V="07/2004" /></ROW><ROW ACTION="ADDUPDATE"
TBL="METERREAD"><METERID V="BXT00_RES5" /><MANUFACTURER V="METERSRUS" /
><SERIALNO V="12345" /><UNINUMBER V="54321" /><METERREADTIME V="07/30/2004
00:00:00" /><BILDETERMCODE V="1" /><STOPREADING V="1500" /><METERREADMONTH V="07/
2004" /></ROW><ROW ACTION="ADDUPDATE" TBL="METERREAD"><METERID V="BXT00_GS1" /
><MANUFACTURER V="METERSRUS" /><SERIALNO V="12345" /><UNINUMBER V="54321" /
><METERREADTIME V="07/30/2004 00:00:00" /><BILDETERMCODE V="1" /><STOPREADING
V="1500" /><METERREADMONTH V="07/2004" /></ROW><ROW ACTION="ADDUPDATE"
TBL="METERREAD"><METERID V="BXT00_GS2" /><MANUFACTURER V="METERSRUS" /><SERIALNO
V="12345" /><UNINUMBER V="54321" /><METERREADTIME V="07/30/2004 00:00:00" /
><BILDETERMCODE V="1" /><STOPREADING V="1500" /><METERREADMONTH V="07/2004" /></
ROW></ADDUPDATE></BABATCH></LOCALMEMORY></ADAPTER>

2007-05-24 13:21:58.053 - The rule returned successfully.
2007-05-24 13:21:58.083 - Output created at C:\Training\Output\out_31324.XML
2007-05-24 13:21:58.083 - Output conversion complete.
2007-05-24 13:21:58.103 - RT_FilePortal.ProcessFile Finished
=====
```

# Part Three

---

## COM Interfaces

Part Three describes the COM interfaces available with the Oracle Utilities Energy Information Platform, and contains the following chapters:

- **Chapter 15: Energy Information Platform COM Interfaces**
- **Chapter 16: Data Source Interface**
- **Chapter 17: Energy Information Platform Database Import Interface**
- **Chapter 18: Oracle Utilities Rules Language and Analysis Interface**
- **Chapter 19: Energy Information Platform Work Queues Interface**
- **Chapter 20: Energy Information Platform Interval Data Interfaces**

---



# Chapter 15

---

## Energy Information Platform COM Interfaces

This chapter provides a broad overview of the Energy Information Platform Component Object Model (COM) interfaces, including a description of how the COM interfaces function, and the format in which the specific COM interface methods/functions are described in later chapters of this manual.

# The Energy Information Platform COM Interfaces

The Energy Information Platform COM interfaces are divided into two layers. The first is a COM Interface level and the second is the functional layer.

## COM Interface Level Functions

The COM Interface level:

1. Accepts calls from the external applications (such as an EAI Adapter),
2. Exposes some of the Energy Information Platform functions as public COM objects,
3. Returns possible errors, and requested data to the calling programs resulting from the execution of these functions,
4. Calls the Energy Information Platform function passing the XML argument sent from the external application. The function will perform the connect and disconnect from the database if required.
5. Other work to satisfy the business event.

## Return Values

An XML structure will be returned to the calling application, from the COM Interface, that contains function execution results. These results will contain one or more of the following:

- Return codes,
- Error codes,
- Descriptions of errors,
- Requested data, and/or
- Other information satisfying the caller's intention.

## Energy Information Platform Level Functions

The Energy Information Platform Level functions/methods are part of several different interfaces, including:

- **DataSource:** The Data Source (IDataSource) interface provides methods related to data source operations, including opening/close database connections, and beginning, committing, and rolling back transactions.
- **Database:** The Database interface (IDatabase) provides methods related to retrieving, updating, and modifying data in the Oracle Utilities Data Repository, including Add/Update/Delete Data, and Interval Data Information Request.
- **Analysis:** The Analysis interface (IAnalysis) provides methods related to billing calculations and analysis, including Cancel/Rebill, Current/Final Bill, Trial Bill, and processing of rate schedules.
- **AREngine:** The AREngine interface (IAREngine) provides methods related to financial management functionality including updating the subledger, journal balancing, and account balancing.
- **Billing:** The Billing interface (IBilling) includes provides related to financial management billing functionality including posting and canceling of charges and bills.
- **Remittance:** The Remittance interface (IRemittance) provides methods related to financial management remittance functionality including posting and processing of payments, batch payments, and payment files.
- **Maintenance:** The Maintenance interface (IMaintenance) provides methods related to financial management maintenance functionality including posting and canceling of adjustments, transfers, write-offs, and refunds.
- **Messaging:** The Messaging interface (IMessaging) provides methods related to messaging functionality including posting, retrieving, removing, listing of messages.
- **Activity Implementations:** The Activity Implementation interface (IActivityImplementation) provides methods for working with activity implementations in WorkflowExpert.
- **Process Versions:** The Process Versions interface (IProcessVersions) provides methods for working with processes and process versions in WorkflowExpert.
- **Process Instance:** The Process Instance interface (IProcessInstance) provides methods for working with process and activity instances in WorkflowExpert.

The specific methods/functions of each interface are described in the Chapters 3 through 14.

## How the Energy Information Platform COM Interfaces Work

All of the Energy Information Platform COM interfaces work in the same manner. The Energy Information Platform COM Interfaces accept an XML argument(s) based on the specific business functions being performed. The Energy Information Platform COM API initially logs on to (or connects to) the database using the data in the `xmlDataSource` parameter. If successful, it then calls an Energy Information Platform function, passing to it the XML string it requires (all Energy Information Platform functions accept XML arguments). The function parses the XML string and, if successful, begins processing the string and processing the data. If the parsing is unsuccessful, the interface will populate an appropriate return parameter with error codes, and a descriptive text string stating that there was an XML parsing error. This information is returned from the Energy Information Platform function. Transactions may encompass the entire job, or may only encompass each Energy Information Platform record in the XML string. If a transaction is unsuccessful, then the COM Layer will populate the appropriate result argument. The return structure will contain descriptive information about why the record(s) erred.

# The Energy Information Platform COM Interface Format

The descriptions of the methods of the Energy Information Platform COM interfaces in this document all follow the same format. This format is as follows:

**Overview:** A brief description and overview of the intended purpose and use of the interface/method.

**Method, Interface, and Syntax:** The method (function) name, interface object, DLL name, Program ID (progid), and syntax for each of the methods available in the interface. This section also includes a brief description of each method.

**Interface Arguments:** Definitions/descriptions of the arguments used by the interface.

**Input Values, including:**

- **Input Data Type Definitions (DTD):** A DTD or XML schema that defines the input data of each argument used by the interface.
- **Input XML Examples:** An example XML string for each argument applicable to the interface.
- **Input Element Descriptions:** Descriptions of the data elements of each argument accepted by the interface.

**Return Values (if any), including:**

- **Return DTD:** A DTD or XML schema that defines the return data of each argument issued by the interface.
- **Return XML:** An example return XML string for each argument applicable to the interface.
- **Return Element Definitions:** Descriptions of the data elements of each argument returned by the interface.

Note that not all methods use all of the above. For instance, many of the Oracle Utilities Receivables Component methods do not have return values.

## Basic Syntax

The basic syntax for Energy Information Platform COM interface methods is as follows:

**Method Name:** <BusinessFunction>

**Interface:** <InterfaceName>

**DLL Name:** <DLL\_NAME.DLL>

**Program ID:** <progid>

**Syntax:**

```
HRESULT <BusinessFunction>(
 [in] BSTR xmlDataSource,
 [in] BSTR xml<BusinessFunction>,
 [in] BSTR other xml arguments*
 [out] BSTR xml<BusinessFunction>Result);
```

- The BusinessFunction is the name of the Energy Information Platform function to satisfy the business event. For example: LODESTAR\_IMPORT.
- The InterfaceName is the name of the interface that contains the method. For example: IAnalysis.
- The DLL Name is the name of the Dynamic-Link Library (DLL) that contains the interface object. For example: LSDB.DLL.
- The Program ID is the program id (progid) of the interface. For example: LSACCT.Billing.
- The xmlDataSource argument is an XML string that contains database connection and other related information (see below for more information).
- The xml<BusinessFunction> argument is an XML string containing all of the information necessary for the execution of the function supporting the business event. The XML and DTDs are defined in the appropriate section of this document for each exposed function. The string 'BusinessFunction' will be replaced by the name of the exposed COM method or function (i.e. xmlLodestarImport).
- In the optional "out" argument, 'BusinessFunction>Result' will reflect the name of the called method, with the string 'Result' concatenated to it (i.e. xmlLodestarImportResult).
- The optional 'other xml arguments' are any other arguments to be passed to the interface in an XML string.

These naming conventions are optional but recommended.

# Common Argument Definitions

## xmlDataSource

All of the Energy Information Platform interfaces use the xmlDataSource argument. The xmlDataSource argument is an XML string that contains database connection and other related information. The Data Type Definition (DTD), an xml example, and element descriptions for the xmlDataSource argument are provided below.

### DTD - xmlDataSource

```
<!DOCTYPE DataSource
[
<!ELEMENT DataSource (Name, ConnectString?, UserID?, Password ?,
 Qualifier?)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT ConnectString (#PCDATA)>
<!ELEMENT UserID (#PCDATA)>
<!ELEMENT Password (#PCDATA)>
<!ELEMENT Qualifier (#PCDATA)>
]>
```

### XML Example - xmlDataSource

```
<DATASOURCE>
 <NAME>LSDATA</NAME>
 <CONNECTSTRING>UID=user1; PWD=user1pass;DSN=LSDATA</CONNECTSTRING>
 <USERID>user1</USERID>
 <PASSWORD>user1pass</PASSWORD>
 <QUALIFIER>PWRLINE</QUALIFIER>
</DATASOURCE>
```

### Element Descriptions - xmlDataSource

Each of the data elements used by the xmlDataSource argument is described below.

**NAME:** The name of the datasource.

**CONNECTSTRING:** The ODBC connect string for the Oracle Utilities Data Repository.

**USERID:** The user ID of the user making the connection to the datasource.

**PASSWORD:** The user's password.

**QUALIFIER:** The qualifier or schema used for database object access (tables, views, etc.).

The only required argument for the DataSource constructors is the data source name. The Qualifier argument provided for each unique data source will become the default for that data source, but can be overridden at any time by explicitly providing them.

## Date Formats

Unless specifically noted otherwise, all Energy Information Platform COM interface methods use the following Date and Date-Time formats:

### Date

Date elements should be in the following format:

**YYYY-MM-DD**

where YYYY is the year, MM is the month, and DD is the day.

### Date-Time

Date-time elements should be in the following format:

**YYYY-MM-DDThh:mm:ss**

where YYYY is the year, MM is the month, DD is the day, hh is the hour, mm is the minute, and ss is the second.



# Chapter 16

---

## Data Source Interface

This chapter describes the methods/functions available to external systems through the Data Source interface (IDataSource). These methods allow users to perform data source operations, such as opening/closing datasource connections from external systems. These functions include the following:

- Open Connection
- Close Connection
- Begin Transaction
- Commit Transaction
- Rollback Transaction
- Execute Query

# Methods, Interface, and Syntax

The methods, interface object, and syntax for the DataSource interface are as follows:

## Open Connection

**Description:** Used to open a persistent database connection in a two-tier client-server (thick or dedicated client) environment.

**Method Name:** OpenConnection

**Interface:** IDataSource

**DLL Name:** LSDB.DLL

**Program ID:** LSDB.DataSource

**Syntax:**

```
HRESULT OpenConnection([in] BSTR xmlDataSource);
```

## Close Connection

**Description:** Used to close a persistent database connection in a two-tier client-server (thick or dedicated client) environment.

**Method Name:** CloseConnection

**Interface:** IDataSource

**DLL Name:** LSDB.DLL

**Program ID:** LSDB.DataSource

**Syntax:**

```
HRESULT CloseConnection([in] BSTR xmlDataSource);
```

## BeginTransaction

**Description:** Used in a two-tier client-server environment to begin a transaction when the client code must encapsulate two or more database access methods into a single transaction.

**Method Name:** BeginTransaction

**Interface:** IDataSource

**DLL Name:** LSDB.DLL

**Program ID:** LSDB.DataSource

**Syntax:**

```
HRESULT BeginTransaction([in] BSTR xmlDataSource);
```

## CommitTransaction

**Description:** Used in a two-tier client-server environment to commit a transaction when the client code must encapsulate two or more database access methods into a single transaction.

**Method Name:** CommitTransaction

**Interface:** IDataSource

**DLL Name:** LSDB.DLL

**Program ID:** LSDB.DataSource

**Syntax:**

```
HRESULT CommitTransaction([in] BSTR xmlDataSource);
```

## RollbackTransaction

**Description:** Used in a two-tier client-server environment to rollback a transaction when the client code must encapsulate two or more database access methods into a single transaction.

**Method Name:** RollbackTransaction

**Interface:** IDataSource

**DLL Name:** LSDB.DLL

**Program ID:** LSDB.DataSource

**Syntax:**

```
HRESULT RollbackTransaction([in] BSTR xmlDataSource);
```

## ExecuteQuery

**Description:** Used to perform one or more commands against the data source. A single command may return records from the data source, insert a new record into the data source, or update or delete an existing record in the data source.

**Method Name:** ExecuteQuery

**Interface:** IDataSource

**DLL Name:** LSDB.DLL

**Program ID:** LSDB.DataSource

**Syntax:**

```
HRESULT ExecuteQuery([in] BSTR xmlDataSource,
 [in] BSTR xmlQueryIn,
 [out, retval] BSTR* xmlQueryOut);
```

## Argument Definitions

The DataSource Interface uses the following arguments:

### **xmlDataSource Argument**

The xmlDataSource argument is an xml string that contains database connection and other related information. A DTD, xml example, and data element descriptions for this argument can be found on page 15-7.

### **xmlQueryIn Argument**

The xmlQueryIn argument is an xml string that specifies a SQL query used to perform one or more commands against the datasource. A DTD, an XML example, and data element descriptions for this argument can be found on page 16-7.

### **xmlQueryOut Argument**

The xmlQueryOut argument is an xml string that contains return values from the Execute Query function. A DTD, an XML example, and data element descriptions for this argument can be found on page 16-9.

# Input Values

The Data Type Definition (DTD), an xml example, and data element descriptions used as input values for the DataSource interface (IDataSource) are provided below.

## xmlDataSource

### DTD - xmlDataSource

```
<!DOCTYPE DataSource
[
<!ELEMENT DataSource (Name, ConnectString?, UserID?, Password ?,
 Qualifier?)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT ConnectString (#PCDATA)>
<!ELEMENT UserID (#PCDATA)>
<!ELEMENT Password (#PCDATA)>
<!ELEMENT Qualifier (#PCDATA)>
]>
```

### XML Example - xmlDataSource

```
<DATASOURCE>
 <NAME>LSDATA</NAME>
 <CONNECTSTRING>Data Source=TS92;User
ID=ls470bx;Password=password;LSProvider=ODP</CONNECTSTRING>
 <USERID>user1 </USERID>
 <QUALIFIER>PWRLINE</QUALIFIER>
</DATASOURCE>
```

### Element Descriptions - xmlDataSource

Each of the data elements used by the xmlDataSource argument is described below.

**NAME:** The name of the datasource.

**CONNECTSTRING:** The connect string for the Oracle Utilities Data Repository, in one of the following formats:

For Oracle databases:

```
<CONNECTSTRING>Data Source=<data_source>;User
ID=<user_id>;Password=<password>;LSProvider=ODP;</CONNECTSTRING>
```

For Microsoft SQL databases:

```
<CONNECTSTRING>Data Source=<address>;Initial
Catalog=<SQL_database>;User
Id=<user_id>;Password=<password>;MultipleActiveResultSets=True;LSP
rovider=MSSQL;</CONNECTSTRING>
```

where:

<data\_source> is the Oracle TNS Name for the data source, from the TNS\_NAMES.ora file (typically located in the \\<machine>\oracle\network\admin directory)

<user\_id> is the user ID for the database connection

<password> is the password for the supplied user ID.

<address> is the IP address or Hostname of the MS SQL Server database server

<SQL\_database> is the name of the MS SQL Server database

**USERID:** The user ID of the user making the connection to the datasource.

**QUALIFIER:** The qualifier or schema used for database object access (tables, views, etc.).

The only required argument for the DataSource constructors is the data source name. The Qualifier argument provided for each unique data source will become the default for that data source, but can be overridden at any time by explicitly providing them.

## xmlQueryIn

### DTD - xmlQueryIn

```
<!DOCTYPE QUERY
[
<!ELEMENT QUERY (COMMAND+)>
<!ELEMENT COMMAND (REQUEST, RECORDSET?, ERROR?)>
<!ATTLIST COMMAND
 ID CDATA #IMPLIED
 NORECORDSCDATA#IMPLIED
 PAGESIZECDATA#IMPLIED
 PAGENOCDATA #IMPLIED >
<!ELEMENT REQUEST (#PCDATA)>
<!ELEMENT RECORDSET (RECORD*)>
<!ATTLIST RECORDSET
 NUMRECORDSCDATA#IMPLIED>
<!ELEMENT RECORD (EMPTY)>
<!ATTLIST RECORD
 COLNAME1CDATA#IMPLIED
 .
 .
 .
>
<!ELEMENT ERROR (#PCDATA)>
]>
```

### XML Example - xmlQueryIn

```
<QUERY>
 <COMMAND ID="" NORECORDS="FALSE" PAGESIZE="128" PAGENO="1">
 <REQUEST>select accountid,uidaccount,customerid,uidcustomer from
PWRLINE.ACCOUNT,pwrline.customer where ACCOUNTID like '%RM%';</REQUEST>
 </COMMAND>
</QUERY>
```

### Element Descriptions - xmlQueryIn

The use of each individual attribute and element in the xmlQueryIn argument is described below.

**QUERY:** The query that contains one or more command elements.

**COMMAND:** The command element(s) that contains specific attributes of the command

Attributes:

**ID:** Optional ID (used to allow for easy identification of the command on return),

**NORECORDS:** Optional. If set to TRUE, the command will not return any records.

**PAGESIZE:** Optional. Used with PAGENO to specify a limited number of records to be returned starting at a specified page number given the specified page size (typically the SQL "order by" clause should be used when paging). Regardless of whether paging is used or not, a single command is limited to returning a maximum of 128 records..

**PAGENO:** Optional. Used with PAGESIZE to specify a limited number of records to be returned starting at a specified page number given the specified page size (typically the SQL "order by" clause should be used when paging). Regardless of whether paging is used or not, a single command is limited to returning a maximum of 128 records.

Elements

**REQUEST:** The actual text of the command.

**RECORDSET:** Optional. Used to return any requested records.

Attributes:

**NUMRECORDS:** Indicates how many records are returned.

**RECORD:** Individual record in the recordset.

Attributes:

**COLNAME:** Name of non-null column in the returned record.

**ERROR:** Description of any error(s) that occurred while processing the command.



---

# Return Values

## xmlQueryOut

### DTD - xmlQueryOut

Same as the xmlQueryIn argument.

### \XML Example - xmlQueryOut

```
<QUERY>
 <COMMAND ID="" NORECORDS="FALSE" PAGESIZE="128" PAGENO="1">
 <REQUEST>select accountid,uidaccount,customerid,uidcustomer from
PWRLINE.ACCOUNT,pwrline.customer where ACCOUNTID like '%RM%';</REQUEST>
 <RECORDSET NUMRECORDS="1">
 <RECORD ACCOUNTID="RM_1001" UIDACCOUNT= "487" CUSTOMERID= "RM_1"
UIDCUSTOMER= "344">
 </RECORD>
 </RECORDSET>
 </COMMAND>
 </QUERY>
```

### Element Descriptions - xmlQueryOut

Same as the xmlQueryIn argument.



# Chapter 17

---

## Energy Information Platform Database Import Interface

This chapter describes a method/function available to external systems through the Energy Information Platform Database Import interface (LSImportDB). This method allows users to import relational data into the Oracle Utilities Data Repository.

## Methods, Interface, and Syntax

The methods, interface objects, and syntax for the Energy Information Platform Database Import interface are as follows:

### Import Data

**Description:** Used to add, update, and/or delete data in the Oracle Utilities Data Repository.

**Method Name:** Import

**Interface:** LSIImportDB

**DLL Name:** LSIImportDB.dll

**Program ID:** LSIImportDB.LSIImportDB

**Syntax:**

```
HRESULT Import ([in] BSTR xmlDataSource,
 [in] VARIANT xmlImport,
 [out] VARIANT xmlImportResult);
```

## Interface Arguments

The Energy Information Platform Database Import interface uses the following arguments:

### **xmlDataSource Argument**

The xmlDataSource argument is an xml string that contains database connection and other related information. A DTD, xml example, and data element descriptions for this argument can be found on page 15-7.

### **xmlImport Argument**

The xmlImport argument contains the appropriate data to be added/updated/deleted. This may be a URL (String/BSTR), a Request object (in an ASP page), an IStream, SAFEARRAY of bytes (VT\_ARRAY|VT\_UI1), a DOMDocument object, or any object that supports IStream, ISequentialStream, or IPersistStream. A DTD, xml examples, and data element descriptions for this argument can be found on page 17-4.

### **xmlImportResult Argument**

The xmlImportResult argument is an xml string that contains return values from the Import function. This argument is a superset of the xmlImport argument. The records that were successful will be removed from the XML. All the records that error out will be present in the output XML along with the error message. An xml example, and data element descriptions for this argument can be found on pages 17-6.

## Input Values

The Data Type Definition (DTD), xml examples, and data element descriptions used as input values for the Energy Information Platform Database interface (LSImportDB) are provided below.

### xmlImport

#### DTD - xmlImport

```
<!DOCTYPE LODESTAR_IMPORT
[
<!ELEMENT LODESTAR_IMPORT (ROWS*)>
<!ATTLIST LODESTAR_IMPORT
 ACTION#PCDATA,
 BATCHSIZE#PCDATA,
 ONERROR#PCDATA>
<!ELEMENT ROWS(ROW*)>
<!ATTLIST ROWS
 TBL #PCDATA>
<!ELEMENT ROW (DBCOLUMNNAME)>
<!ATTLIST ROW
 TBL CDATA #IMPLIED,
 ACTION(ADD|UPDATE|DELETE|ADDUPDATE)#IMPLIED>
<!ELEMENT DBCOLUMNNAME(#PCDATA)>
<!ATTLIST DBCOLUMNNAME
 V CDATA "column value",
 NV CDATA "new column value">
]>
```

#### XML Examples - xmlImport

##### Add Example

```
<LODESTAR_IMPORT ACTION="ADD" BATCHSIZE="50" ONERROR="CONTINUE">
 <ROWS>
 <ROW TBL="ACCOUNT">
 <CUSTOMERID V="Cust1"/>
 <ACCOUTNID V="Acct1"/>
 <STARTTIME V="01/01/2000 00:00:00"/>
 </ROW>
 </ROWS>
</LODESTAR_IMPORT>
```

##### Update Example

```
<LODESTAR_IMPORT ACTION="UPDATE" BATCHSIZE="50" ONERROR="STOP">
 <ROWS TBL="ACCOUNT">
 <ROW>
 <CUSTOMERID V="Cust1"/>
 <ACCOUTNID V="Acct1" NV="Acct100"/>
 <STARTTIME V="01/01/2001 00:00:00"/>
 </ROW>
 </ROWS>
</LODESTAR_IMPORT>
```

## Element Descriptions - xmlImport

**LODESTAR\_IMPORT:** Root element containing one or more sets of records, each defined in a ROWS element.

Attributes:

**ACTION:** Defines the action performed. Possible values are:

- **ADD:** Adds one or more records to the database. It is an error if the record(s) is already present.
- **UPDATE:** Updates one or more existing records in the database. It is an error if the record does not exist.
- **DELETE:** Deletes one or more records in the database. It is an error if the record to be deleted does not exist.
- **ADDUPDATE:** Adds or Updates one or more records in the database. This is the Default behavior. If a record is present, the function updates it. If the record does not exist, the function adds the record.

**BATCHSIZE:** The number of records to be sent to the database in a single call. Larger numbers generally improve performance. The default is 50. The maximum is 500.

**ONERROR:** Defines how errors are handled. Possible values include:

- **CONTINUE:** Continue importing records after error.
- **STOP:** Stop importing records when an error occurs.

Elements:

**ROWS:** Element containing one or more records for import, each defined in a ROW element.

Attributes:

**TBL:** Table name for all the records defined in the ROW elements included in the ROW element.

Elements:

**ROW:** Element containing one record for import.

Attributes:

**TBL:** Table name for the record. If present in a ROW element, this overrides the TBL attribute on the ROWS element.

Elements:

**DBCOLUMNNAME:** Actual database name of the column. For example, when importing records in the Account ID column of the Account table, this would be "ACCOUNTID".

Attributes:

**V:** Column value. If not present, NULL is assumed. If the value is an XML, it will be specified as a child of the DBCOLUMNNAME element.

**NV:** New column value for the record. Must be specified if updating a key column.

**Notes:**

- The "NV" (New value) attribute is only required for identity columns that need to be updated.
- Also, for "DELETE" operations, providing just the identity columns is sufficient.
- Any missing "V" (Value) attribute will be treated as a NULL value for the column.

## Return Values

This section describes the data returned from the Energy Information Platform Database interface.

### xmlImportResult

#### XML Example - xmlImportResult

```
<LODESTAR_IMPORT ACTION="ADD" BATCHSIZE="50" ONERROR="CONTINUE">
 <ROWS_IMPORTED>10</ROWS_IMPORTED>
 <ROWS_FAILED>1</ROWS_FAILED>
 <ROWS>
 <ROW TBL="ACCOUNT" ERROR="Unable to find customer Cust1">
 <CUSTOMERID V="Cust1"/>
 <ACCOUNID V="Acct1"/>
 <STARTTIME V="01/01/2000 00:00:00"/>
 </ROW>
 </ROWS>
</LODESTAR_IMPORT>
```

#### Element Descriptions - xmlImportResult

The xmlImportResult argument contains the same elements as the xmlImport argument, with the following additions:

**ROWS\_IMPORTED:** The number of records successfully imported.

**ROWS\_FAILED:** The number of records that failed import.

**ROWS:** Element containing one or more records for import, each defined in a ROW element (same as for xmlImport).

Elements:

**ROW:** Element containing a record that failed import.

Attributes:

**TBL:** Table name for the record that failed import.

**ERROR:** Error message explaining why the record failed import.



# Chapter 18

---

## Oracle Utilities Rules Language and Analysis Interface

This chapter describes the methods/functions available to external systems through the Oracle Utilities Rules Language and Analysis interfaces. These methods allow users to run pre-defined, pre-existing rate schedules and to execute current/final and bill correction processing.

# Methods, Interface, and Syntax

## Rules Language Interface Methods

The methods, interface objects, and syntax for the Rules Language interface are as follows:

### Run Rate

**Description:** Used to synchronously run a pre-defined rate schedule.

**Method Name:** RunRate

**Interface:** RunRs

**DLL Name:** LSRF.DLL

**Program ID:** LSRF.RunRs

**Syntax:**

```
HRESULT RunRate ([in] BSTR xmlRulesSchedule,
 [retval] BSTR xmlSaveToXML);
```

### Run Rate Queued

**Description:** Used to asynchronously run a pre-defined rate schedule. Rate schedules run using this method are queued, and executed by the Energy Information Platform Report Framework.

**Method Name:** RunRateQueued

**Interface:** RunRs

**DLL Name:** LSRF.DLL

**Program ID:** LSRF.RunRs

**Syntax:**

```
HRESULT RunRateQueued ([in] BSTR xmlRulesSchedule,
 [retval] BSTR GUID);
```

### Run Rate Status

**Description:** Used to obtain the status of a rate schedule initiated by the Run Rate Queued method.

**Method Name:** RunRateStatus

**Interface:** RunRs

**DLL Name:** LSRF.DLL

**Program ID:** LSRF.RunRs

**Syntax:**

```
HRESULT RunRateStatus ([in] BSTR xmlDataSource,
 [in] BSTR GUID);
[retval] BSTR Status);
```

### Run Rate Result

**Description:** Used to obtain results from a rate schedule initiated by the Run Rate Queued method. This method can only be used with rate schedules with a Status of “DONE”.

**Method Name:** RunRateResult

**Interface:** RunRs

**DLL Name:** LSRF.DLL

**Program ID:** LSRF.RunRs

**Syntax:**

```
HRESULT RunRateResult ([in] BSTR xmlDataSource,
 [in] BSTR GUID);
[retval] BSTR xmlSaveToXML);
```

## Run Rate Stop

**Description:** Used to terminate an already running rate schedule initiated by the Run Rate Queued method.

**Method Name:** RunRateStop

**Interface:** RunRs

**DLL Name:** LSRF.DLL

**Program ID:** LSRF.RunRs

**Syntax:**

```
HRESULT RunRateStop ([in] BSTR xmlDataSource,
 [in] BSTR GUID);
```

## Run Rate Error

**Description:** Used to obtain the error message from rate schedules initiated by the Run Rate Queued method with a Status of “ERROR”.

**Method Name:** RunRateError

**Interface:** RunRs

**DLL Name:** LSRF.DLL

**Program ID:** LSRF.RunRs

**Syntax:**

```
HRESULT RunRateError ([in] BSTR xmlDataSource,
 [in] BSTR GUID,
 [retval] BSTR Error);
```

## Analysis Interface Methods

The methods, interface objects, and syntax for the Analysis interface are as follows:

### Run Analysis

**Description:** Used to synchronously execute billing in either Current/Final or Cancel/Rebill (bill correction) modes.

**Method Name:** RunAnalysis

**Interface:** Analysis

**DLL Name:** LSRF.DLL

**Program ID:** LSRF.Analysis

**Syntax:**

```
HRESULT RunRate ([in] BSTR xmlAnalysis,
 [retval] BSTR xmlAnalysisResult);
```

### Run Analysis Queued

**Description:** Used to asynchronously execute billing in either Current/Final or Cancel/Rebill (bill correction) modes. Billing processes run using this method are queued, and executed by the Energy Information Platform Report Framework.

**Method Name:** RunAnalysisQueued

**Interface:** Analysis

**DLL Name:** LSRF.DLL

**Program ID:** LSRF.Analysis

**Syntax:**

```
HRESULT RunRateQueued ([in] BSTR xmlAnalysis,
 [retval] BSTR GUID);
```

### Run Analysis Status

**Description:** Used to obtain the status of a billing process initiated by the Run Analysis Queued method.

**Method Name:** RunAnalysisStatus

**Interface:** Analysis

**DLL Name:** LSRF.DLL

**Program ID:** LSRF.Analysis

**Syntax:**

```
HRESULT RunRateStatus ([in] BSTR xmlDataSource,
 [in] BSTR GUID);
[retval] BSTR Status);
```

## Run Analysis Result

**Description:** Used to obtain results from a billing process initiated by the Run Analysis Queued method. This method can only be used with billing processes with a Status of “DONE”.

**Method Name:** RunAnalysisResult

**Interface:** Analysis

**DLL Name:** LSRF.DLL

**Program ID:** LSRF.Analysis

**Syntax:**

```
HRESULT RunRateResult ([in] BSTR xmlDataSource,
 [in] BSTR GUID);
[retval] BSTR xmlAnalysisResult);
```

## Run Analysis Stop

**Description:** Used to terminate an already running billing process initiated by the Run Analysis Queued method.

**Method Name:** RunAnalysisStop

**Interface:** Analysis

**DLL Name:** LSRF.DLL

**Program ID:** LSRF.Analysis

**Syntax:**

```
HRESULT RunRateStop ([in] BSTR xmlDataSource,
 [in] BSTR GUID);
```

## Run Analysis Error

**Description:** Used to obtain the error message from rbilling processes initiated by the Run Analysis Queued method with a Status of “ERROR”.

**Method Name:** RunAnalysisError

**Interface:** Analysis

**DLL Name:** LSRF.DLL

**Program ID:** LSRF.Analysis

**Syntax:**

```
HRESULT RunRateError ([in] BSTR xmlDataSource,
 [in] BSTR GUID,
 [retval] BSTR Error);
```

## Interface Arguments

The functions/methods of the Oracle Utilities Rules Language and Analysis interfaces use the following arguments:

### xmlDataSource Argument

The xmlDataSource argument is an xml string that contains database connection and other related information. A DTD, xml example, and data element descriptions for this argument can be found on page 15-7 in the *Oracle Utilities Energy Information Platform Configuration Guide*.

### xmlRulesSchedule Argument

The xmlRulesSchedule argument is an xml string that specifies the parameters necessary to run a rate schedule. The elements map directly to “-” parameters for the RUNRS executable file. Details on these settings may be found in **Chapter 8: Setting Up Processing to run in Batch Mode** in the *Oracle Utilities Energy Information Platform Configuration Guide*. The structure is designed to support multiple schedule requests. A DTD, xml example, and data element descriptions for this argument can be found on page 18-8.

### xmlAnalysis Argument

The xmlAnalysis argument is an xml string that specifies the parameters necessary to run billing processing. The elements map directly to “-” parameters for the CURFINAL and BILLCOR executable files. Details on these settings may be found in **Chapter 8: Setting Up Processing to run in Batch Mode** in the *Oracle Utilities Energy Information Platform Configuration Guide*. The structure is designed to support multiple schedule requests. An xml schema, xml example, and data element descriptions for this argument can be found on page 18-11.

### xmlSaveToXML Argument

The xmlSaveToXML argument is an xml string that contains results from any SAVE TO XML statements in the rate schedule. A DTD, xml example, and data element descriptions for this argument can be found on page 18-15.

### xmlAnalysisResult Argument

The xmlAnalysisResult argument is an xml string that contains results from billing processing. An xml schema, xml example, and data element descriptions for this argument can be found on page 18-16.

### GUID

A Unique string identifier for the rate schedule or billing process that was queued. This is returned from the RunRateQueued and RunAnalysisQueued methods, and is an input for the RunRateStatus, RunRateResults, RunRateStop, RunRateError, RunAnalysisStatus, RunAnalysisResults, RunAnalysisStop, and RunAnalysisError methods.

---

## Status

The status of a queued rate schedule or billing process. Will be one of the following:

Status Value	Description
SUSPENDED	The process is suspended
QUEUED	The process is waiting to be started.
INIT	Generation application is in an initialization phase.
PROCESS	The process is currently executing.
DONE	The process is fully completed.
DELETING	The process is waiting to be deleted.
ERROR	The process cannot be completed because of an error.

## Error

The error returned from a rate schedule or billing process whose Status is “ERROR”.

# Input Values

The input values for the methods/functions of the Rules Language and Analysis interfaces (RunRs and Analysis) are provided below.

## xmlRulesSchedule

### DTD - xmlRulesSchedule

```
<!DOCTYPE LODESTAR_RUNSCHEDULE
[
 <!ELEMENT LODESTAR_RUNSCHEDULE (DATASOURCE, CONFIG_FILE_NAME?,
 TIMEOUT?, USERID, SCHEDULE_ARGUMENTS)>
 <!ELEMENT DATASOURCE (NAME, CONNECTSTRING, USERID?, PASSWORD?, QUALIFIER?)>
 <!ELEMENT NAME (#PCDATA)>
 <!ELEMENT CONNECTSTRING (#PCDATA)>
 <!ELEMENT USERID (#PCDATA)>
 <!ELEMENT PASSWORD (#PCDATA)>
 <!ELEMENT QUALIFIER (#PCDATA)>
 <!ELEMENT CONFIG_FILE_NAME (#PCDATA)>
 <!ELEMENT TIMEOUT(#PCDATA)>
 <!ELEMENT USERID(#PCDATA)>
 <!ELEMENT SCHEDULE_ARGUMENTS (INTERVAL_DATABASE_NAME?,
 RATE_FORM_IDENTIFIER, START_DATE, STOP_DATE?, SAVE_RESULTS?,
 (ACCOUNT | CUSTOMER)?, INTERVAL_DATA_ERROR?, RSID_VALUES?)>
 <!ELEMENT INTERVAL_DATABASE_NAME (#PCDATA)>
 <!ELEMENT RATE_FORM_IDENTIFIER (#PCDATA)>
 <!ELEMENT START_DATE (#PCDATA)>
 <!ELEMENT STOP_DATE (#PCDATA)>
 <!ELEMENT SAVE_RESULTS EMPTY>
 <!ELEMENT ACCOUNT (ALL | ID | LIST_NAME)>
 <!ELEMENT CUSTOMER (ALL | ID | LIST_NAME)>
 <!ELEMENT ALL EMPTY>
 <!ELEMENT ID (#PCDATA)>
 <!ELEMENT LIST_NAME (#PCDATA)>
 <!ELEMENT INTERVAL_DATA_ERROR (#PCDATA)>
 <!ELEMENT RSID_VALUES (RSID_STEM*, RSID_VALUE*)>
 <!ELEMENT RSID_STEM (RSID_VALUE+)>
 <ATTLIST RSID_STEM NAME CDATA #REQUIRED>
 <!ELEMENT RSID_VALUE EMPTY>
 <ATTLIST RSID_VALUE
 NAMECDATA #REQUIRED
 TYPE(F|I|S|D) "S"
 VALUECDATA #REQUIRED>
 <!ELEMENT NAME (#PCDATA)>
 <!ELEMENT TYPE (#PCDATA)>
 <!ELEMENT VALUE (#PCDATA)>
]>
```



## XML Example - xmlRulesSchedule

```
<LODESTAR_RUNSCHEDULE>
 <DATASOURCE>
 <NAME>DB1</NAME>
 <CONNECTSTRING>DSN=local; UID=User1; PWD=password ;</CONNECTSTRING>
 <QUALIFIER>TEST</QUALIFIER>
 </DATASOURCE>
 <TIMEOUT>3600</TIMEOUT>
 <USERID>SuperUser</USERID>
 <CONFIG_FILE_NAME> C:\Lodestar\Bin\Lodestar.cfg</CONFIG_FILE_NAME>
 <SCHEDULE_ARGUMENTS >
 <INTERVAL_DATABASE_NAME>RDB </INTERVAL_DATABASE_NAME>
 <RATE_FORM_IDENTIFIER> OPC01:JUR1:RS1:0 </RATE_FORM_IDENTIFIER>
 <START_DATE> 2000/01/01 </START_DATE>
 <STOP_DATE> 2000/01/31 </STOP_DATE>
 <SAVE_RESULTS/>
 <ACCOUNT><ALL/></ACCOUNT>
 <INTERVAL_DATA_ERROR> 1 </INTERVAL_DATA_ERROR>
 <RSID_VALUES>
 <RSID_VALUE NAME="TEST1" TYPE="I" VALUE ="20">
 </ RSID_VALUES >
 </SCHEDULE_ARGUMENTS >
</LODESTAR_RUNSCHEDULE>
```

## Element Descriptions - xmlRulesSchedule

Each of the data elements used in the xmlRulesSchedule argument is described below. The corresponding RUNRS arguments for each element are listed in parentheses.

**LODESTAR\_RUNSCHEDULE:** Root element containing a DATASOURCE and SCHEDULE\_ARGUMENTS element that define the rate schedule to be run.

Elements:

**DATASOURCE:** Element containing connection information for the data source to be accessed by the rate schedule.

Elements:

**NAME:** Name of the data source.

**CONNECTSTRING:** Connection string to the data source, using the following syntax:  
DSN=local; UID=User1; PWD=password;

**QUALIFIER:** Qualifier for the data source.

**TIMEOUT:** Number of seconds after which the process will time out.

**USERID:** User name of the user initiating the rate schedule.

**CONFIG\_FILE\_NAME** (-f): The name of the configuration file that defines the working environment of the Energy Information Platform software (e.g., directs the software where to find and place the application data files and so on). If you do not supply a value for *configfilename*, the system uses the default, which is LODESTAR.CFG. For information about the contents of this configuration file, please refer to the *Oracle Utilities Energy Information Platform Configuration Guide*.

**SCHEDULE\_ARGUMENTS:** Element containing specific arguments and parameters used by the rate schedule.

Elements:

**INTERVAL\_DATABASE\_NAME** (-b): The name of the Interval Database (btrieve file) that contains the interval data for the calculations. If you do not specify a value, the program applies the option associated with the userID. (You can find this value by opening Data Manager or other application program, then select **Tools->Options->Interval Data Source**. It is the value under **Database/Input File Name**.)

**RATE\_FORM\_IDENTIFIER** (-v): The identifier for the rate form. **Required.** OPCO:JUR:RS[VER] are the operating company code, jurisdiction code, rate form code, and optional version number that together identify the rate form. If you do not supply a version number, the program automatically uses the rate form that was applicable on the stop date of the analysis period (see STOP\_DATE). If you do supply a version number, the program uses the trial version indicated (typically used only for testing).

**START\_DATE** (-s): The start date of the date range for the calculations. You can specify the date as an absolute or relative value, as described in **Chapter 8: Setting Up Processing to run in Batch Mode** in the *Oracle Utilities Energy Information Platform Configuration Guide*.

**STOP\_DATE** (-t): The stop date of the date range for the calculations. You can specify the date as an absolute or relative value, as described in **Chapter 8: Setting Up Processing to run in Batch Mode** in the *Oracle Utilities Energy Information Platform Configuration Guide*.

**SAVE\_RESULTS** (-k): Tells the program to save the results, as specified in the rate form. If you leave this switch out of the command, the program ignores all SAVE statements in the rate form (this is useful if you want to debug the rate form before using it in a production mode, for example). If you supply the switch, the program saves interval data, table saves, and CIS records. Since RUNRS does not input an account, there is no saving of determinants to a Bill History record, unless -ai is specified (see below).

**ACCOUNT** (-a): Optional Account All/ID/List name - default is no account. If you specify ID you can save to the Bill History Table.

**CUSTOMER** (-a): Optional Customer All/ID/List name - default is no customer.

Elements: (apply to either ACCOUNT or CUSTOMER)

**ALL:** Specifies whether or not to process all Accounts/Customers. See ACCOUNT/CUSTOMER above.

**ID:** Specifies the Account/Customer ID to process. See ACCOUNT/CUSTOMER above.

**LIST\_NAME:** Specifies the list name of Accounts/Customers to process. See ACCOUNT/CUSTOMER above.

**INTERVAL\_DATA\_ERROR** (-i): Sets interval data handling. Uses the same values as the INTD\_ERROR\_STOP configuration parameter (0,1,2,3). Default # is 1 (error stop). If not set, the **Tools->Options->Error Handling->On Rules Language Interval Data Error** option is used.

**RSID\_VALUES** (-x): Element containing a block of elements defining identifier values to be used for this rate schedule.

Elements:

**RSID\_VALUE** : An individual identifier value to be used for this rate schedule.

Attributes:

**NAME:** The name of the identifier value.

**TYPE:** The type of the identifier value.

**VALUE:** The value for the identifier.

## xmlAnalysis

### Schema - xmlAnalysis

```

<?xml version="1.0" ?>
<xs:schema id="NewDataSet" targetNamespace="http://tempuri.org/Analysis.xsd"
 xmlns:mstns="http://tempuri.org/Analysis.xsd"
 xmlns="http://tempuri.org/Analysis.xsd"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
 attributeFormDefault="qualified" elementFormDefault="qualified">
 <xs:element name="LS_ANALYSIS">
 <xs:complexType>
 <xs:sequence>
 <xs:element name="CONFIG_FILE_NAME" type="xs:string" minOccurs="0" />
 <xs:element name="DATASOURCE" minOccurs="1" maxOccurs="1">
 <xs:complexType>
 <xs:sequence>
 <xs:element name="NAME" type="xs:string" minOccurs="1"
maxOccurs="1" />
 <xs:element name="CONNECTSTRING" type="xs:string" minOccurs="1"
maxOccurs="1" />
 <xs:element name="QUALIFIER" type="xs:string" minOccurs="0"
maxOccurs="1" />
 </xs:sequence>
 </xs:complexType>
 </xs:element>
 <xs:element name="ANALYSIS" minOccurs="0" maxOccurs="1">
 <xs:complexType>
 <xs:sequence>
 <xs:element name="ACCOUNT" minOccurs="0" maxOccurs="1">
 <xs:complexType>
 <xs:sequence>
 <xs:element name="ID" type="xs:string" minOccurs="0"/>
 <xs:element name="LIST" type="xs:string" minOccurs="0"/>
 <xs:element name="ALL" type="xs:boolean" minOccurs="0"/>
 <xs:element name="FILENAME" type="xs:string" minOccurs="0"/>
 </xs:sequence>
 </xs:complexType>
 </xs:element>
 <xs:element name="CUSTOMER" minOccurs="0" maxOccurs="1">
 <xs:complexType>
 <xs:sequence>
 <xs:element name="ID" type="xs:string" minOccurs="0"/>
 <xs:element name="LIST" type="xs:string" minOccurs="0"/>
 <xs:element name="ALL" type="xs:boolean" minOccurs="0"/>
 <xs:element name="FILENAME" type="xs:string" minOccurs="0"/>
 </xs:sequence>
 </xs:complexType>
 </xs:element>
 <xs:element name="USER_SPECIFIED_STOP" type="xs:string"
minOccurs="0" />
 <xs:element name="NO_APPROVE_ACCOUNTS" type="xs:boolean"
minOccurs="0" />
 <xs:element name="NO_RELATED_SUMMARY_ACCOUNTS" type="xs:boolean"
minOccurs="0" />
 <xs:element name="APPROVE_ACCOUNT_PAGE" type="xs:boolean"
minOccurs="0" />
 <xs:element name="FINAL_BILL" type="xs:boolean" minOccurs="0" />
 <xs:element name="CIS_RECORDS" type="xs:string" minOccurs="0" />
 <xs:element name="TRANSACTION_REPORTS" type="xs:string"
minOccurs="0" />
 <xs:element name="LOGGING_CONFIG_FILE" type="xs:string"
minOccurs="0" />
 <xs:element name="LGNA_TABLES" type="xs:boolean" minOccurs="0" />
 <xs:element name="BILL_PERIODS_TO_CORRECT" type="xs:boolean"
minOccurs="0" />

```

```
 <xs:element name="BILL_TYPE" type="xs:string" minOccurs="0" />
 <xs:element name="NO_BH_DATES_TIMES" type="xs:boolean"
minOccurs="0" />
 <xs:element name="NO_BH_DET_VALUES" type="xs:boolean"
minOccurs="0" />
 <xs:element name="REBILL_REASON_CODE" type="xs:string"
minOccurs="0" />
 </xs:sequence>
 <xs:attribute name="TYPE" type="xs:string" />
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

## XML Examples - xmlAnalysis

### Current/Final Bill

```
<?xml version="1.0" encoding="utf-8" ?>
<LS_ANALYSIS>
 <DATASOURCE>
 <NAME>abc</NAME>
 <CONNECTSTRING>DSN=dev;UID=ABC1;PWD=pwd</CONNECTSTRING>
 <QUALIFIER>QUAL1</QUALIFIER>
 </DATASOURCE>
 <CONFIG_FILE_NAME>c:\lodestar\cfg\lodestar.cfg</CONFIG_FILE_NAME>
 <ANALYSIS TYPE="CR">
 <ACCOUNT>
 <LIST>AccountList1</LIST>
 </ACCOUNT>
 <USER_SPECIFIED_STOP>01/01/1999 00:00:00</USER_SPECIFIED_STOP>
 <FINAL_BILL/>
 </ANALYSIS>
</LS_ANALYSIS>
```

### Cancel/ReBill

```
<?xml version="1.0" encoding="utf-8" ?>
<LS_ANALYSIS>
 <DATASOURCE>
 <NAME>abc</NAME>
 <CONNECTSTRING>DSN=dev;UID=ABC1;PWD=pwd</CONNECTSTRING>
 <QUALIFIER>QUAL1</QUALIFIER>
 </DATASOURCE>
 <CONFIG_FILE_NAME>c:\lodestar\cfg\lodestar.cfg</CONFIG_FILE_NAME>
 <ANALYSIS TYPE="BC">
 <CUSTOMER>
 <ID>CUSTOMER1</ID>
 </CUSTOMER>
 <BILL_PERIODS_TO_CORRECT>2</BILL_PERIODS_TO_CORRECT>
 <BILL_TYPE>CR</BILL_TYPE>
 </ANALYSIS>
</LS_ANALYSIS>
```

## Element Descriptions - xmlAnalysis

Each of the data elements used in the xmlAnalysis argument is described below. The corresponding CURFINAL.EXE and BILLCORR.EXE arguments for each element are listed in parentheses.

**LS\_ANALYSIS:** Root element containing a DATASOURCE and ANALYSIS element that define the billing process to be run.

Elements:

**DATASOURCE:** Element containing connection information for the data source to be accessed by the billing process.

Elements:

**NAME:** Name of the data source.

**CONNECTSTRING:** Connection string to the data source, using the following syntax:  
DSN=local; UID=User1; PWD=password;

**QUALIFIER:** Qualifier for the data source.

**CONFIG\_FILE\_NAME** (-f): The name of the configuration file that defines the working environment of the Energy Information Platform software (e.g., directs the software where to find and place the application data files and so on). If you do not supply a value for *configfilename*, the system uses the default, which is LODESTAR.CFG. For information about the contents of this configuration file, please refer to the *Oracle Utilities Energy Information Platform Configuration Guide*.

**ANALYSIS:** Element containing specific arguments and parameters used by the billing process.

Attributes:

**TYPE:** Specifies the type of analysis. Can be either BC (Bill Correction), or CF (Current/Final bill).

Elements:

**ACCOUNT** (-a): Account All/ID/List name/File name. The default is all accounts.

**CUSTOMER** (-a): Customer All/ID/List name/File name. The default is all customers.

Elements: (apply to either ACCOUNT or CUSTOMER)

**ID:** Specifies the Account/Customer ID to process. See ACCOUNT/CUSTOMER above.

**LIST:** Specifies the list name of Accounts/Customers to process. See ACCOUNT/CUSTOMER above.

**ALL:** Specifies whether or not to process all Accounts/Customers. See ACCOUNT/CUSTOMER above.

**FILENAME:** Specifies the path and file name of a file containing Accounts/Customer IDs to process. See ACCOUNT/CUSTOMER above.

**USER\_SPECIFIED\_STOP** (-t, CURFINAL): The User Specified Stop Date, in MM/DD/YYYY format. The default is computed.

**NO\_APPROVE\_ACCOUNTS** (-s): If you include this element, accounts are not approved. The default is to Approve.

**NO\_RELATED\_SUMMARY\_ACCOUNTS** (-m): If you include this element, related summary accounts are not processed. The default is to process related accounts.

**FINAL\_BILL** (-n, CURFINAL): Indicates that the bill is to be a final bill.

**CIS\_RECORDS** (-o): The name of the output file used by the CIS system. The default is LODESTAR.CIS.

**TRANSACTION\_REPORTS** (-w): The filename for transaction reports [default: YYYYMMDD.HHMM].

**LOGGING\_CONFIG\_FILE** (-lcfg): Name of an optional logging configuration file that specifies where error and log messages are sent.

**LGNA\_TABLES** (-o): Indicates that output should be sent to the LGNA tables.

**BILL\_TYPE** (-t, BILLCORR): The Bill type: Cancel/Rebill (CR), Adjustment (A), Cancel (C), Rebill (R). The default is Cancel/Rebill (CR).

**BILL\_PERIODS\_TO\_CORRECT** (-b): The number of bill periods to correct. The default is 0.

**NO\_BH\_DATES\_TIMES** (-h): If you include this element, Bill History dates and times will not be used. The default is to use them. **Note:** This element only applies if dates/times can be computed from interval data.

**NO\_BH\_DET\_VALUES** (-v): If you include this element, current Bill History determinant values will not be used. The default is to use them. **Note:** This element only applies if dates/times can be computed from interval data.

**REBILL\_REASON\_CODE** (-d): The reason for the bill correction.

**APPROVE\_ACCOUNT\_PAGE** (-1): If you include this element, each Account is approved if no error occurs. The default is to approve each group[ of accounts.

# Return Values

## xmlSaveToXML

### DTD - xmlSaveToXML

```
<!DOCTYPE RSID_VALUES
[
<!ELEMENT RSID_VALUES (RSID_STEM*, RSID_VALUE*)>
<!ELEMENT RSID_STEM (RSID_VALUE+)>
<ATTLIST RSID_STEM NAME CDATA #REQUIRED>
<!ELEMENT RSID_VALUE EMPTY>
<ATTLIST RSID_VALUE
 NAMECDATA#REQUIRED
 TYPE(F|I|S|D)"S"
 VALUECDATA#REQUIRED>
]>
```

### XML Example - xmlSaveToXML

```
<RSID_VALUES>
 <RSID_STEM NAME="VAL1">
 <RSID_VALUE NAME="UOM" TYPE="S" VALUE="79"/>
 <RSID_VALUE NAME="D_KWH" TYPE="F" VALUE="1000"/>
 </RSID_STEM>
 <RSID_VALUE NAME="MY_RS_ID" TYPE="S" VALUE="TEST"/>
 <RSID_VALUE NAME="MY_RS_ID2" TYPE="F" VALUE="20.99"/>
</RSID_VALUES>
```

### Element Descriptions - xmlSaveToXML

The use of each individual attribute and element in the xmlLodestarRunScheduleResult argument is described below.

**RSID\_VALUES:** Element containing a block of elements defining identifier values returned from the rate schedule.

Elements:

**RSID\_STEM :** Element specifying a stem identifier and containing one or more RSID\_VALUE elements, each specifying a tail identifier.

Attributes:

**NAME:** The name of the stem identifier.

**RSID\_VALUE :** An individual identifier value return from the rate schedule. When found within a RSID\_STEM element, RSID\_VALUE elements represent tail identifiers.

Attributes:

**NAME:** The name of the identifier value.

**TYPE:** The type of the identifier value.

**VALUE:** The value for the identifier.

## xmlAnalysisResult

### Schema - xmlAnalysisResult

```
<?xml version="1.0"?>
<xs:schema targetNamespace="http://tempuri.org/~vs1DC.xsd"
 xmlns:mstns="http://tempuri.org/~vs1DC.xsd"
 xmlns="http://tempuri.org/~vs1DC.xsd"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
 attributeFormDefault="qualified" elementFormDefault="qualified">
 <xs:element name="LS_ANALYSIS">
 <xs:complexType>
 <xs:sequence>
 <xs:element name="RESULT" minOccurs="0" maxOccurs="1">
 <xs:complexType>
 <xs:attribute name="CODE" type="xs:string" />
 </xs:complexType>
 </xs:element>
 </xs:sequence>
 <xs:attribute name="TYPE" type="xs:string" />
 </xs:complexType>
 </xs:element>
</xs:schema>
```

### XML Example - xmlAnalysisResult

```
<?xml version="1.0" encoding="utf-8"?>
<LS_ANALYSIS TYPE="BC">
 <RESULT CODE="99" />
</LS_ANALYSIS>
```

### Element Descriptions - xmlAnalysisResult

The use of each individual attribute and element in the xmlAnalysisResult argument is described below.

**LS\_ANALYSIS:** Element containing the result of the billing process.

Attributes:

**TYPE:** Specifies the type of billing process. Can be either BC (Bill Correction), or CF (Current/Final bill).

Elements:

**RESULT :** Element containing result code for billing process.

Attributes:

**CODE:** The result code of the billing process. Can be one of the following:

- 0: All accounts that could be billed were billed successfully.
- 1: Analysis could not start.
- 99: At least one account failed to bill.



# Chapter 19

---

## Energy Information Platform Work Queues Interface

This chapter describes the methods/functions available to external systems through the Energy Information Platform Work Queues interface (IWorkQueue). These methods allow users to perform a number of work queue functions, including:

- Open
- Assign
- Unassign
- Update
- Resolve
- Approve
- Reject
- Close
- Reopen
- Exists

# Methods, Interfaces, and Syntax

The methods, interface objects, and syntax for the Work Queues interface are as follows:

## Open

**Description:** Used to open a work queue item.

**Method Name:** Open

**Interface:** IWorkQueue

**DLL Name:** LSWorkQueue.DLL

**Program ID:** LSWorkQueue.WorkQueue

**Syntax:**

```
HRESULT Open([in] BSTR xmlDataSource,
 [in] BSTR xmlWQItem,
 [out, retval] BSTR* xmlWQItemOut);
```

## Assign

**Description:** Used to assign a work queue item to a specified user.

**Method Name:** Assign

**Interface:** IWorkQueue

**DLL Name:** LSWorkQueue.DLL

**Program ID:** LSWorkQueue.WorkQueue

**Syntax:**

```
HRESULT Assign([in] BSTR xmlDataSource,
 [in] BSTR xmlWQItem,
 [in] BSTR strUserId,
 [in, optional] BSTR strNote);
```

## Unassign

**Description:** Used to unassigned a previously assigned work queue item.

**Method Name:** Unassign

**Interface:** IWorkQueue

**DLL Name:** LSWorkQueue.DLL

**Program ID:** LSWorkQueue.WorkQueue

**Syntax:**

```
HRESULT Unassign([in] BSTR xmlDataSource,
 [in] BSTR xmlWQItem,
 [in] BSTR strUserId,
 [in, optional] BSTR strNote);
```

## Update

**Description:** Used to update a specified attribute of an open work queue item.

**Method Name:** Update

**Interface:** IWorkQueue

**DLL Name:** LSWorkQueue.DLL

**Program ID:** LSWorkQueue.WorkQueue

**Syntax:**

```
HRESULT Update([in] BSTR xmlDataSource,
 [in] BSTR xmlWQItem,
 [in] BSTR strUserId,
 [in, optional] BSTR strNote);
```

**Note:** only Queue Code, Priority Level, WorkBytime or Data elements can be updated.

## Resolve

**Description:** Used to resolve an open work queue item.

**Method Name:** Resolve

**Interface:** IWorkQueue

**DLL Name:** LSWorkQueue.DLL

**Program ID:** LSWorkQueue.WorkQueue

**Syntax:**

```
HRESULT Resolve([in] BSTR xmlDataSource,
 [in] BSTR xmlWQItem,
 [in] BSTR strUserId,
 [in, optional] BSTR strNote);
```

## Approve

**Description:** Used to approve an open work queue item.

**Method Name:** Approve

**Interface:** IWorkQueue

**DLL Name:** LSWorkQueue.DLL

**Program ID:** LSWorkQueue.WorkQueue

**Syntax:**

```
HRESULT Approve([in] BSTR xmlDataSource,
 [in] BSTR xmlWQItem,
 [in] BSTR strUserId,
 [in, optional] BSTR strNote);
```

## Reject

**Description:** Used to reject an open work queue item.

**Method Name:** Reject

**Interface:** IWorkQueue

**DLL Name:** LSWorkQueue.DLL

**Program ID:** LSWorkQueue.WorkQueue

**Syntax:**

```
HRESULT Reject([in] BSTR xmlDataSource,
```

```
[in] BSTR xmlWQItem,
[in] BSTR strUserId,
[in, optional] BSTR strNote);
```

## Close

**Description:** Used to close an open work queue item.

**Method Name:** Close

**Interface:** IWorkQueue

**DLL Name:** LSWorkQueue.DLL

**Program ID:** LSWorkQueue.WorkQueue

**Syntax:**

```
HRESULT Close([in] BSTR xmlDataSource,
 [in] BSTR xmlWQItem,
 [in] BSTR strUserId,
 [in, optional] BSTR strNote);
```

## Reopen

**Description:** Used to reopen a previously closed work queue item.

**Method Name:** Reopen

**Interface:** IWorkQueue

**DLL Name:** LSWorkQueue.DLL

**Program ID:** LSWorkQueue.WorkQueue

**Syntax:**

```
HRESULT Reopen([in] BSTR xmlDataSource,
 [in] BSTR xmlWQItem,
 [in] BSTR strUserId,
 [in, optional] BSTR strNote);
```

## Exists

**Description:** Used to determine if a specified work queue item exists.

**Method Name:** Exists

**Interface:** IWorkQueue

**DLL Name:** LSWorkQueue.DLL

**Program ID:** LSWorkQueue.WorkQueue

**Syntax:**

```
HRESULT Exists([in] BSTR xmlDataSource,
 [in] BSTR xmlWQItem,
 [out, retval] BOOL bExists);
```

## Interface Arguments

The methods available in the Work Queues interface use the following arguments:

### **xmlDataSource Argument**

The xmlDataSource argument is an xml string that contains database connection and other related information. A DTD, xml example, and data element descriptions for this argument can be found on page 15-7.

### **xmlWQItem Argument**

The xmlWQItem argument is an xml string that contains a work queue item. Xml examples and data element descriptions for this argument can be found on page 19-6.

### **xmlWQItemOut Argument**

The xmlWQItemOut argument is an xml string that contains a work queue item. An xml example and data element descriptions for this argument can be found on page 19-13.

### **strUserId Argument**

The strUserID argument is a string that contains the user id of the user invoking the method.

### **strNote Argument**

The strNote argument is a string that contains an optional note related to the work queue item.

### **bExists Argument**

The bExists argument is a Boolean value that indicates if a specified work queue item exists or not. A value of 1 indicates the item exists, a value of 0 (zero) indicate the item does not exist. This argument is returned from the Exists method only.

## Input Values

Xml examples and data element descriptions of the xmlWQItem argument used as an input value for the Work Queues interface (IWorkQueue) are provided below.

### xmlWQItem XML Elements

```
<WQITEM>
 <UID/>
 <TYPE/>
 <QUEUE/>
 <PRIORITYLEVEL/>
 <WORKBYTIME/>
 <OPENEDTIME/>
 <OPENEDBYUSERID/>
 <OPENEDNOTE/>
 <ASSIGNEDTOUSERID/>
 <ASSIGNEDTOTIME/>
 <APPRSTEPNUM/>
 <APPRSTEPCOUNT/>
 <APPRLEVEL/>
 <APPROVALONLY/>
 <APPROVED/>
 <PRODUCT/>
 <PROCESSNAME/>
 <PROCESSID/>
 <UIDRATEFORM/>
 <READDATE/>
 <ACCOUNTID/>
 <AMOUNT/>
 <ITEMDATA/>
 <CHANNELNO/>
 <INTDSTARTTIME/>
 <JURISCODE/>
 <METERID/>
 <OPCOCODE/>
 <RECORDERID/>
 <TRANSACTIONNO/>
</WQITEM>
```

### XML Examples - xmlWQItem

#### Open

```
<WQITEM>
 <UID/>
 <TYPE/>
 <TYPE VALTYPE="NEW">REGULAR</TYPE>
 <QUEUE/>
 <PRIORITYLEVEL/>
 <WORKBYTIME/>
 <OPENEDBYUSERID>Karen</OPENEDBYUSERID>
 <OPENEDNOTE>This item is not "Approval Only"</OPENEDNOTE>
 <ASSIGNEDTOUSERID/>
 <ASSIGNEDTOTIME/>
 <APPRSTEPNUM/>
 <APPRSTEPCOUNT/>
 <APPROVALONLY/>
 <ACCOUNTID>8000029</ACCOUNTID>
 <AMOUNT>100000.00</AMOUNT>
 <CHANNELNUM>1</CHANNELNUM>
 <INTDSTARTTIME>2002-12-06T15:08:10</INTDSTARTTIME>
 <ITEMDATA>Warning: bill Exceeds $100000.00</ITEMDATA>
 <JURISCODE>SF</JURISCODE>
 <METERID">25</METERID>
 <OPCOCODE/>
```

```

<OPCOCODE>GECO</OPCOCODE>
<PROCESSID>Steven</PROCESSID>
<READDATE>2002-12-10</READDATE>
<RECORDERID>2002</RECORDERID>
<TRANSACTIONNO>4561</TRANSACTIONNO>
<UIDRATEFORM>444</UIDRATEFORM>
</WQITEM>

```

## Assign

For assigning an item to a new user:

```

<WQITEM>
 <UID>160</UID>
 <TYPE> Code_1</TYPE>
 <QUEUE> Queue_1</QUEUE>
 <PRIORITYLEVEL>1</PRIORITYLEVEL>
 <WORKBYTIME>2002-11-25T12:44:46</WORKBYTIME>
 <ASSIGNEDTOUSERID/>
 <ASSIGNEDTOTIME/>
 <APPRSTEPNUM>1</APPRSTEPNUM>
 <APPRSTEPCOUNT>0</APPRSTEPCOUNT>
 <APPRLEVEL>2</APPRLEVEL>
 <ASSIGNEDTOUSERID VALTYPE="NEW">Steven</ASSIGNEDTOUSERID>
 <APPROVALONLY>FALSE</APPROVALONLY>
</WQITEM>

```

For reassigning an item from a previous user to a new user:

```

<WQITEM>
 <UID>160</UID>
 <TYPE> Code_1</TYPE>
 <QUEUE> Queue_1</QUEUE>
 <PRIORITYLEVEL>1</PRIORITYLEVEL>
 <WORKBYTIME>2002-11-25T12:44:46</WORKBYTIME>
 <ASSIGNEDTOUSERID>Karen</ASSIGNEDTOUSERID>
 <ASSIGNEDTOTIME>2002-11-25T12:00:00</ASSIGNEDTOTIME>
 <APPRSTEPNUM>1</APPRSTEPNUM>
 <APPRSTEPCOUNT>0</APPRSTEPCOUNT>
 <APPRLEVEL>2</APPRLEVEL>
 <ASSIGNEDTOUSERID VALTYPE="NEW">Steven</ASSIGNEDTOUSERID>
 <APPROVALONLY>FALSE</APPROVALONLY>
</WQITEM>

```

## Unassign

```

<WQITEM>
 <UID>20</UID>
 <TYPE> Code_1</TYPE>
 <QUEUE>TORINO_WQ</QUEUE>
 <PRIORITYLEVEL>1</PRIORITYLEVEL>
 <WORKBYTIME>2002-11-07T16:53:02</WORKBYTIME>
 <ASSIGNEDTOUSERID>karen</ASSIGNEDTOUSERID>
 <ASSIGNEDTOTIME>2002-11-07T14:54:46</ASSIGNEDTOTIME>
 <APPRSTEPNUM>1</APPRSTEPNUM>
 <APPRSTEPCOUNT>0</APPRSTEPCOUNT>
 <APPRLEVEL>2</APPRLEVEL>
 <ASSIGNEDTOUSERID VALTYPE="NEW" />
 <APPROVALONLY>FALSE</APPROVALONLY>
</WQITEM>

```

## Update

```

<WQITEM>
 <UID>142</UID>
 <TYPE>Code_1</TYPE>
 <QUEUE>Queue_1</QUEUE>
 <PRIORITYLEVEL>1</PRIORITYLEVEL>
 <WORKBYTIME>2002-11-22T12:53:14</WORKBYTIME>

```

```
<ASSIGNEDTOUSERID/>
<ASSIGNEDTOTIME/>
<APPRSTEPNUM/>
<APPRSTEPCOUNT/>
<QUEUE VALTYPE="NEW"> TORINO_WQ </QUEUE>
<PRIORITYLEVEL VALTYPE="NEW">2</PRIORITYLEVEL>
<WORKBYTIME VALTYPE="NEW">2002-11-22T11:54:00</WORKBYTIME>
</WQITEM>
```

## Resolve

```
<WQITEM>
<UID>160</UID>
<TYPE> Code_1</TYPE>
<QUEUE> Queue_1</QUEUE>
<PRIORITYLEVEL>1</PRIORITYLEVEL>
<WORKBYTIME>2002-11-25T12:44:46</WORKBYTIME>
<ASSIGNEDTOUSERID/>
<ASSIGNEDTOTIME/>
<APPRSTEPNUM/>
<APPRSTEPCOUNT/>
<APPRLEVEL/>
<APPROVALONLY/>
</WQITEM>
```

## Approve

```
<WQITEM>
<UID>62</UID>
<TYPE> Code_1</TYPE>
<QUEUE> Queue_1</QUEUE>
<PRIORITYLEVEL>2</PRIORITYLEVEL>
<WORKBYTIME>2002-11-19T10:19:07</WORKBYTIME>
<ASSIGNEDTOUSERID>Steven</ASSIGNEDTOUSERID>
<ASSIGNEDTOTIME>2002-11-18T10:19:07</ASSIGNEDTOTIME>
<APPRSTEPNUM>1</APPRSTEPNUM>
<APPRSTEPCOUNT>0</APPRSTEPCOUNT>
<APPRLEVEL>Level_Code_1</APPRLEVEL>
<APPROVALONLY/>
</WQITEM>
```

## Reject

```
<WQITEM>
<UID>160</UID>
<TYPE> Code_1</TYPE>
<QUEUE> Queue_1</QUEUE>
<PRIORITYLEVEL>1</PRIORITYLEVEL>
<WORKBYTIME>2002-11-19T10:19:07</WORKBYTIME>
<ASSIGNEDTOUSERID>Steven</ASSIGNEDTOUSERID>
<ASSIGNEDTOTIME>2002-11-18T10:19:07</ASSIGNEDTOTIME>
<APPRSTEPNUM>1</APPRSTEPNUM>
<APPRSTEPCOUNT>0</APPRSTEPCOUNT>
<APPRLEVEL>Level_Code_1</APPRLEVEL>
<APPROVALONLY/>
</WQITEM>
```

## Close

```
<WQITEM>
<UID>52</UID>
<TYPE> Code_1</TYPE>
<QUEUE> Queue_1</QUEUE>
<PRIORITYLEVEL>1</PRIORITYLEVEL>
<WORKBYTIME>2002-11-14T00:00:00</WORKBYTIME>
<OPENEDTIME>2002-11-13T13:31:16</OPENEDTIME>
<ASSIGNEDTOUSERID/>
<ASSIGNEDTOTIME/>
<APPRSTEPNUM/>
```



```
<APPRSTEPCOUNT/>
<APPRLEVEL/>
<APPROVALONLY/>
</WQITEM>
```

## Reopen

```
<WQITEM>
<UID>50</UID>
<TYPE> Code_1</TYPE>
<QUEUE> Queue_1</QUEUE>
<PRIORITYLEVEL>1</PRIORITYLEVEL>
<ASSIGNEDTOUSERID/>
<ASSIGNEDTOTIME/>
<APPRSTEPNUM/>
<APPRSTEPCOUNT/>
<APPRLEVEL/>
<APPROVALONLY/>
</WQITEM>
```

## Exists

```
<WQITEM>
<UID>161</UID>
<TYPE> Code_1</TYPE>
<QUEUE> Queue_1</QUEUE>
<PRIORITYLEVEL>1</PRIORITYLEVEL>
<WORKBYTIME>2002-11-27T10:57:36</WORKBYTIME>
<ASSIGNEDTOUSERID/>
<ASSIGNEDTOTIME/>
<APPRSTEPNUM>1</APPRSTEPNUM>
<APPRSTEPCOUNT>1</APPRSTEPCOUNT>
<APPRLEVEL>Level_Code_1</APPRLEVEL>
<ASSIGNEDTOUSERID/>
<APPROVALONLY/>
</WQITEM>
```

## Element Descriptions - xmlWQItem

Each of the data elements used by the xmlWQItem argument is described below.

Elements:

**UID:** A unique identifier for the work queue item.

**TYPE:** The work queue type for the item. This is the only attribute that is required when opening a new work queue item. Supports the VALTYPE="NEW" attribute for new work queue items.

**QUEUE:** The work queue for the item. If not explicitly provided then the default queue of the corresponding type will be used. Supports the VALTYPE="NEW" attribute when updating work queue items.

**PRIORITYLEVEL:** The priority level of the work queue item. If not explicitly provided then the default priority level of the corresponding type will be used. Supports the VALTYPE="NEW" attribute when updating work queue items.

**WORKBYTIME:** (Open items only) This is the optional time by which the item is expected to be resolved. If not provided then it is calculated from the Default Work By Hours values of the corresponding type. Supports the VALTYPE="NEW" attribute when updating work queue items.

**OPENEDTIME:** The time when the item was opened. This is automatically determined when the item is opened and can never be subsequently changed.

**OPENEDNOTE:** An optional note for the item, created when the item was opened.

**ASSIGNEDTOUSERID:** (Opened items only) UserId of user to which the item is currently assigned. This will change each time the item is unassigned, reassigned, resolved, approved, rejected, or closed. Requires the VALTYPE="NEW" attribute when updating work queue items.

**ASSIGNEDTOTIME:** (Opened items only) Time at which item was assigned to current user. This will also change each time the item is unassigned, reassigned, resolved, approved, rejected, or closed. Requires the VALTYPE="NEW" attribute when updating work queue items.

**APPRSTEPNUM:** (Opened items only) Current approval step of corresponding approval process that item is on. This will change as the item move through the approval process.

**APPRSTEPCOUNT:** (Opened items only) Current number of approvals for this approval step.

**APPRLEVEL:** (Opened items only) Approval level corresponding to current approval step described above.

**APPROVALONLY:** Provided by corresponding type. If true, indicates that the item requires a final approved/rejected state.

**APPROVED:** Final approved/rejected state for approval only items.

**PRODUCT:** Optional product associated with the item - used for filtering (i.e. Oracle Utilities Billing Component). If not explicitly provided then the default product of the corresponding type will be used, else it is NULL. If provided, the PRODUCT must have a corresponding record in the LODESTAR Product table.

**PROCESSNAME:** Optional process name associated with the item - used for filtering (i.e. AutoBill). If provided, the PRODUCTNAME must have a corresponding record in the Business Process table.

**PROCESSID:** Optional identifying information that can be used to distinguish one instance of a process from another.

**UIDRATEFORMVERSION:** Optional rate form version from which item was generated.

**READDATE:** Optional read date of current billing cycle from which item was generated. Will be used together with the Stop Billing flag similar to how ACCOUNTNOTES are used.

**UIDACCOUNT:** Optional. Unique identifier for the account associated with the work queue item.

**ACCOUNTID:** Optional. Account ID for the account associated with the work queue item.

**AMOUNT:** Optional. An amount associated with the work queue item.

**DATA:** Optional additional data specific to the item, represented as an XML structure. Supports the VALTYPE="NEW" attribute when updating work queue items.

## Specifying New Values

When updating work queue items via the Update, Assign, or Unassign methods, elements for values to be updated should be listed twice, first with the current value, followed by the new value. The element containing the new value should also include the VALTYPE attribute (with a value of "NEW").

For updating an item: In the example below, Priority Level is changed from 1 to 2, and the Work By Time is changed from 2002-11-22T12:53:14 to 2002-11-22T11:54:00.

```
<WQITEM>
 <UID>142</UID>
 <TYPE>Code_1</TYPE>
 <QUEUE>Queue_1</QUEUE>
 <PRIORITYLEVEL>1</PRIORITYLEVEL>
 <WORKBYTIME>2002-11-22T12:53:14</WORKBYTIME>
 <ASSIGNEDTOUSERID/>
 <ASSIGNEDTOTIME/>
 <APPRSTEPNUM/>
 <APPRSTEPCOUNT/>
 <QUEUE VALTYPE="NEW"> TORINO_WQ </QUEUE>
 <PRIORITYLEVEL VALTYPE="NEW">2</PRIORITYLEVEL>
 <WORKBYTIME VALTYPE="NEW">2002-11-22T11:54:00</WORKBYTIME>
</WQITEM>
```

For assigning an item to a new user:

```
<WQITEM>
 <UID>160</UID>
 <TYPE> Code_1</TYPE>
 <QUEUE> Queue_1</QUEUE>
 <PRIORITYLEVEL>1</PRIORITYLEVEL>
 <WORKBYTIME>2002-11-25T12:44:46</WORKBYTIME>
 <ASSIGNEDTOUSERID/>
 <ASSIGNEDTOTIME/>
 <APPRSTEPNUM>1</APPRSTEPNUM>
 <APPRSTEPCOUNT>0</APPRSTEPCOUNT>
 <APPRLEVEL>2</APPRLEVEL>
 <ASSIGNEDTOUSERID VALTYPE="NEW">Steven</ASSIGNEDTOUSERID>
 <APPROVALONLY>FALSE</APPROVALONLY>
</WQITEM>
```

For reassigning an item from a previous user to a new user:

```
<WQITEM>
 <UID>160</UID>
 <TYPE> Code_1</TYPE>
 <QUEUE> Queue_1</QUEUE>
 <PRIORITYLEVEL>1</PRIORITYLEVEL>
 <WORKBYTIME>2002-11-25T12:44:46</WORKBYTIME>
 <ASSIGNEDTOUSERID>Karen</ASSIGNEDTOUSERID>
 <ASSIGNEDTOTIME>2002-11-25T12:00:00</ASSIGNEDTOTIME>
 <APPRSTEPNUM>1</APPRSTEPNUM>
 <APPRSTEPCOUNT>0</APPRSTEPCOUNT>
 <APPRLEVEL>2</APPRLEVEL>
 <ASSIGNEDTOUSERID VALTYPE="NEW">Steven</ASSIGNEDTOUSERID>
 <APPROVALONLY>FALSE</APPROVALONLY>
</WQITEM>
```

## Custom Parameter Values

In addition to the elements listed, work queue items may have additional custom parameter values explicitly provided when opened. These items must have corresponding records in the Work Queue Custom Parameter table, and must match the column names in the Work Queue Open Item and Work Queue Closed Item tables in the Oracle Utilities Data Repository.

## Return Values

The data returned from the Work Queues interfaces is described in the following xml example, and data element descriptions.

### xmlWQItemOut

The xmlWQItemOut argument is the same as the xmlWQItem argument.

#### Element Descriptions - xmlWQItemOut

The data elements for the xmlWQItemOut argument are the same as the xmlWQItem argument.



# Chapter 20

---

## Energy Information Platform Interval Data Interfaces

This chapter describes the methods/functions available to external systems through the Energy Information Platform Interval Data interfaces (LSINTD). These interfaces allow users to work with interval data stored in the Oracle Utilities Data Repository. This chapter includes:

- **LSINTD Interface Methods, and Parameter Descriptions**
- **Interval Data Source Providers**
- **Sample Application using LSINTD**
- **INTDVB Compatibility**

# LSINTD Interface Methods, and Parameter Descriptions

This section details the interfaces, methods and parameters in the published Energy Information Platform COM interval data interfaces.

## IIntDDataSource Interface

The IIntDDataSource object uses the provided connection string sent to the connect method to connect an interval datasource. A data source is defined as an interval data storage media, and may be RDB, BTE, LSE or XML. The IIntDDataSource object has a number of methods which may be called. Published interfaces appear in bold face and detailed below.

### Interface Definition

```
[
 odl,
 uuid(D4F2C360-E3E3-41f0-AF1F-0708F863D437),
 version(1.0),
 dual,
 oleautomation
]
interface IIntDDataSource : IDispatch {
[id(1), helpstring("method Connect")]
HRESULT Connect(BSTR ConnectString);
[id(2), helpstring("method Flush")]
HRESULT Flush();
[propget, id(3), helpstring("property Recorders")]
HRESULT Recorders([out, retval] IIntDRecorderList* *pVal);
[propget, id(4), helpstring("property Cuts")]
HRESULT Cuts([out, retval] IIntDCutList* *pVal);
[id(5), helpstring("method LoadDates")]
HRESULT LoadDates([in] BSTR Recorder,
 [in] long Channel,
 [in] DATE StartDate,
 [in] DATE StopDate,
 [out, retval] IIntDCut** RetVal);
[id(6), helpstring("method GetCutByKey")]
HRESULT GetCutByKey([in] BSTR Recorder,
 [in] long Channel,
 [in] DATE StartDate,
 [out, retval] IIntDPureCut** RetVal);

[id(7), helpstring("method GetCutByUID")]
HRESULT GetCutByUID([in] BSTR UID, [out, retval] IIntDPureCut** RetVal);

[id(8), helpstring("method GetImporter")]
HRESULT GetImporter([out, retval] IIntDImporter** RetVal);
};
```

This interface has properties and methods that are not published. The published and user-available functions are listed below.

### Published Method: Connect

The IIntDDataSource object uses the provided connection string to determine which data provider must be used, loads it dynamically, and forwards the connection string to it.

The Connect method accepts one argument called "ConnectionString". This argument defines a provider which will be used to connect to an interval data source. This argument also contains parameters specific for the current session which define interval data behavior. The published providers are described later in this document.

The ConnectString argument is defined by the following XML format:

```
<INTDATASOURCE>
 <PROVIDER PROGID="provider's ProgId">
```



```

 . . . provider-specific parameters . . . (such as RDB-DataSource XML or BTE
filename)
 </PROVIDER>
 <SESSION>
 . . . connection-specific parameters . . .
 </SESSION>
</INTDATASOURCE>

```

The **PROGID** attribute of the **PROVIDER** element defines a ProgID of a COM component that implements a particular interval data source provider. This COM component must support the internal IIntDProvider interface.

The **SESSION** element contains session-specific parameters. This element may contain settings for the current session and are provider-independent. This means that they affect generic cut functions and behavior and not data access behavior. The session is defined in detail below. The following XML format is used to define the session:

```

<SESSION>
 [<QualityCode>Character</QualityCode>]
 [<DayStart>Integer</DayStart>]
 [<INTD_NEW_JOIN [INTDMERGESELECT="RECENT|AVERAGE|BEST|MAXIMUM"]
[CHECKVALID="Y|N"] />]
 [<CHECK_UOM_MATCHING [CHECK_COMPATIBILITY="Y|N"] />]
 [<DO_NOT_USE_UOM_FOR_SCALING/>]
</SESSION>

```

The elements used within the SESSION element include:

- **QualityCode:** Ignore interval data values when status code of value is worse than this QualityCode. Valid values for this element are [A-Z] | [0-9], space, or no value. If no value is present then a status code of space is used. If the QualityCode element is not present, the default used is a value of 8. If the status code of an interval is worse than the character specified in the QualityCode code element, then that interval will be returned as a missing interval (Value of 0, Status code '9').
- **DayStart:** This is used as the start of the day. Valid values are 0 - 11, where 0 is for Midnight, 1 is for 1 A.M., 2 is for 2 A.M., up to 11 for 11 A.M. If DayStart element is not present, the default value is 0 (Midnight).
- **INTD\_NEW\_JOIN:** Determines the criteria when merging interval data cuts. See **Interval Data Merge** on page 2-14 of the *Data Manager User's Guide* for more information.
- **CHECK\_UOM\_MATCHING:** If this element exists, the interval data component checks UOM Compatibility when merging cuts across a specified date range; by default the component does not check UOM compatibility. If **CHECK\_COMPATIBILITY** is 'Y' component will check the UOM's compatibility (matching).
- **DO\_NOT\_USE\_UOM\_FOR\_SCALING:** If this element exists, interval data is totalized when scaling or aggregating, regardless of the Totalize Method flag specified in the Unit-of-Measure table.

#### Example 1 (QualityCode only)

```

<SESSION>
 <QualityCode></QualityCode> (a space is assumed)
</SESSION>

```

#### Example 2 (Use defaults)

```

<SESSION/>

```

**Example 3**

```
<SESSION>
 <QualityCode>7</QualityCode>
 <DayStart></DayStart> (This starts the day at 12:00 AM)
 <CHECK_UOM_MATCHING CHECK_COMPATIBILITY="Y"/> (Do not allow merging cuts
with incompatible UOM's)
</SESSION>
```

**Published Method: LoadDates**

This method creates and loads an IIntDCut object from the then-active provider (data store) for a user-specified date range using all settings specified in the Session parameter.

```
HRESULT LoadDates (
 [in] BSTR Recorder,
 [in] BSTR Channel,
 [in] DATE StartDate,
 [in] DATE StopDate,
 [out, retval] IIntDCut** pRetVal);
```

This method accepts 4 arguments and returns a pointer to 1 object.

- **Recorder:** The Recorder BSTR may contain a maximum of 64 bytes.
- **Channel:** The channel BSTR may contain a positive value of less than 32767.
- **StartDate:** StartDate is a Date Value representing the date of the beginning of the range to expect interval data to be returned.
- **StopDate:** StopDate is a Date Value that indicates the end time of the interval data request.
- **pRetVal:** A pointer to the cut object (IIntDCut) created by the method over the specified date range will be returned.

**Published Method: GetCutByKey**

This method loads an IIntDPureCut object from the then-active provider (data store) for a user-specified date range using all settings specified in the Session parameter.

The parameters to this method detail a full cut key which uniquely identifies an Interval Data Cut.

```
[id(6), helpstring("method GetCutByKey")]
HRESULT GetCutByKey([in] BSTR Recorder,
 [in] long Channel,
 [in] DATE StartDate,
 [out, retval] IIntDPureCut** RetVal);
```

This method accepts 3 arguments and returns pointers to 2 objects.

- **Recorder:** The Recorder BSTR may contain a maximum of 64 bytes.
- **Channel:** The channel BSTR may contain a positive value of less than 32767.
- **StartDate:** StartDate is a Date Value representing the date of the requested cut.
- **Property: Recorders:** A pointer to a newly created IIntDRecorderList object is returned to the calling program.
- **Property: Cuts:** A pointer to a newly created IIntdCutList object which contains all cuts is returned to the calling program.

## IIntDRecorderList Interface

The IIntDRecorderList interface provides basic navigation through the list of IIntDRecorder. It extends the COM interface IEnumVARIANT that provides ability to use “for each” statements in Visual Basic and JScript.

### Interface Definition

```
[
 object,
 uuid(77F1D0B4-D5A6-472F-B0AB-855739021B3A),
 dual,
 helpstring("IIntDRecorderList Interface"),
 pointer_default(unique)
]
interface IIntDRecorderList : IDispatch
{
 [propget, id(1), helpstring("property Count")]
 HRESULT Count([out, retval] long *pVal);
 [id(2), helpstring("method Add")]
 HRESULT Add([in] BSTR RecorderId, [out, retval] IIntDRecorder **pVal);
 [propget, id(3), helpstring("property Item")]
 HRESULT Item([in] BSTR RecorderId, [out, retval] IIntDRecorder* *pVal);
 [id(4), helpstring("method Remove")]
 HRESULT Remove(BSTR RecorderId);
 [id(5), helpstring("method RemoveAll")]
 HRESULT RemoveAll();
 [id(0xffffffffc), propget, restricted, hidden, helpstring("property _newEnum")]
 HRESULT _newEnum([out, retval] IUnknown* *pVal);
};
```

**Property: \_newEnum** (Interface IEnumVARIANT): This property is implemented by using rules of the COM environment, and provides “for each” functionality.

**Property: Count:** Contains list length.

**Property: Item:** Returns IIntDRecorder object for the particular RecorderID.

### Method: Add

Inserts the new recorder ID into a recorder list and returns a newly created IIntDRecorder object.

### Method: Remove

Removed recorder from a recorders list.

### Method: RemoveAll

Removes all recorders from a recorders list.

## IIntDRecorder Interface

The IIntDRecorder interface provides access to an IIntDRecorder object.

### Interface Definition

```
[
 object,
 uuid(864AB91F-1B89-44DB-8C91-32BDE4965673),
 dual,
 helpstring("IIntDRecorder Interface"),
 pointer_default(unique)
]
interface IIntDRecorder : IDispatch
{
 [propget, id(1), helpstring("property RecorderId")]
 HRESULT RecorderId([out, retval] BSTR *pVal);
 [propget, id(2), helpstring("property Channels")]
 HRESULT Channels([out, retval] IIntDChannelList* *pVal);
};
```

**Property: RecorderId:** This property contains a recorder ID.

**Property: Channels:** This property returns an IIntDChannelList object.

## IIntDChannelList Interface

The IIntDChannelList interface provides basic navigation through the list of IIntDChannel. It extends the COM interface IEnumVARIANT that provides ability to use “for each” statements in Visual Basic and JScript.

### Interface Definition

```
[
 object,
 uuid(9D6893A1-95A5-4153-BF93-FB2B7CBE23D1),
 dual,
 helpstring("IIntDChannelList Interface"),
 pointer_default(unique)
]
interface IIntDChannelList : IDispatch
{
 [propget, id(1), helpstring("property Count")]
 HRESULT Count([out, retval] long *pVal);
 [id(2), helpstring("method Add")]
 HRESULT Add([in] short ChannelNum, [out, retval] IIntDChannel
 **pVal);
 [propget, id(3), helpstring("property Item")]
 HRESULT Item([in] short ChannelNum, [out,retval] IIntDChannel*
 *pVal);
 [id(4), helpstring("method Remove")]
 HRESULT Remove([in] short ChannelNum);
 [id(5), helpstring("method RemoveAll")]
 HRESULT RemoveAll();
 [id(0xfffffff), propget, restricted, hidden, helpstring("property _newEnum")]
 HRESULT _newEnum([out, retval] IUnknown* *pVal);
};
```

**Property: \_newEnum** (Interface IEnumVARIANT): This property is implemented by using rules of the COM environment, and provides “for each” functionality.

**Property: Count:** Contains a list length.

**Property Item:** Returns an IIntDChannel object for the particular channel number.

**Method: Add**

Inserts a new channel into the channel list and returns a newly created IIntDChannel object

**Method: Remove**

Removes a particular channel from the channel list.

**Method: RemoveAll**

Removes all channels from the channel list.

## IIntDChannel Interface

The IIntDChannel interface provides access to an IIntDChannel object.

**Interface Definition**

```
[
 object,
 uuid(F6B8C61B-C92D-41AE-AD42-377D7F29D776),
 dual,
 helpstring("IIntDChannel Interface"),
 pointer_default(unique)
]
interface IIntDChannel : IDispatch
{
 [propget, id(1), helpstring("property RecorderId")]
 HRESULT RecorderId([out, retval] BSTR *pVal);
 [propget, id(2), helpstring("property ChannelNum")]
 HRESULT ChannelNum([out, retval] short *pVal);
 [propget, id(3), helpstring("property Cuts")]
 HRESULT Cuts([out, retval] IIntDCutList/*IUnknown*/ *pVal);
};
```

**Property: RecorderId:** Contains a recorder name.

**Property: ChannelNum:** Contains a channel number.

**Property: Cuts:** Returns an IIntDCutSeries object.

## IIntDCutSeries Interface

The IIntdCutSeries interface provides basic navigation through the list of IIntDPureCut objects. It extends the COM interface IEnumVARIANT that provides ability to use “for each” statements in Visual Basic and JScript.

### Interface Definition

```
[
 object,
 uuid(08E6B6BF-9EC6-4F3C-B4D0-374A968FADA7),
 dual,
 helpstring("IIntdCutSeries Interface"),
 pointer_default(unique)
]
interface IIntDCutSeries : IDispatch {
 [id(0xffffffffc), propget, restricted, hidden]
 HRESULT _NewEnum([out, retval] IUnknown** ppUnk);

};
```

**Property: \_newEnum** (Interface IEnumVARIANT): This property is implemented by using rules of the COM environment, and provides “for each” functionality.

## IIntDCutList Interface

The IIntdCutList interface extends the IIntDCutSeries interface.

### Interface Definition

```
[
 object,
 uuid(0C6126A6-523E-4D95-A7CA-DE326F7656A2),
 dual,
 helpstring("IIntDCutList Interface"),
 pointer_default(unique)
]
interface IIntDCutList : IIntDCutSeries
{
 [propget, id(1), helpstring("property Count")]
 HRESULT Count([out, retval] long *pVal);
 [id(2), helpstring("method Add")]
 HRESULT Add([in] IIntDPureCut *Cut, [out, retval] IIntDPureCut
 **pVal);
 [id(3), helpstring("method RemoveAll")]
 HRESULT RemoveAll();
};
```

**Property: Count:** Contains list length.

### Method: Add

Inserts new a pure cut into a cut list and returns a newly created IIntDPureCut object. In addition, if the data source supports auditing, an archived copy of the pure cut will also be created.

### Method: RemoveAll

Removes all cuts from cut list.

## IIntDCutMessageList Interface

The IIntDCutMessageList interface provides basic navigation through the list of cut messages.

### Interface Definition

```
[
 object,
```

```
 uuid(64E86328-13FB-4E22-8D5D-88A2024B09E3),
 dual,
 helpstring("IIntDCutMessageList Interface"),
 pointer_default(unique)
]
 interface IIntDCutMessageList : IDispatch
 {
 [id(0xffffffffc), propget, restricted, hidden, helpstring("property _newEnum")]
 HRESULT _newEnum([out, retval] IUnknown* *pVal);
 [id(1), helpstring("method AddMessage")]
 HRESULT AddMessage([in] BSTR Message);
 };
```

**Property \_newEnum** (Interface IEnumVARIANT): This property is implemented by using rules of the COM environment, and provides “for each” functionality.

#### **Method: AddMessage**

This method adds a new message into a message list.

## **IIntDCutMessage Interface**

The IIntDMessage interface provides access to an IIntDMessage object.

#### **Interface Definition**

```
[
 object,
 uuid(D2D37216-2B87-49A7-A4E9-10D4227ED2BD),
 dual,
 helpstring("IIntDCutMessage Interface"),
 pointer_default(unique)
]
interface IIntDCutMessage : IDispatch
{
 [propget, id(1), helpstring("property Message")]
 HRESULT Message([out, retval] BSTR *pVal);
};
```

**Property: Message:** Contains a message string.

## IntervalList Interface

The IIntervalList interface provides basic navigation through a list of interval indexes. Indexes are zero based.

### Interface Definition

```
[
 object,
 uuid(ECFE6996-143A-4a96-8A91-514887053CF2),
 dual,
 helpstring("IIntervalList Interface"),
 pointer_default(unique)
]
interface IIntervalList : IDispatch
{
 [propget, id(1), helpstring("property Count")]
 HRESULT Count([out, retval] long *pVal);
 [id(2), helpstring("method Item")]
 HRESULT Item([in] long Index, [out, retval] long *IntervalIndex);
};
```

**Property: Count:** Contains list length.

### Method: Item

This method returns an interval index from particular list item.

## IIntDPureCut Interface

The IIntDPureCut interface provides access to an IIntDPureCut object.

### Interface Definition

```
[
 object,
 uuid(4495B4E5-3AED-4EFB-A447-4AD92D5EF2AF),
 dual,
 helpstring("IIntDPureCut Interface"),
 pointer_default(unique)
]
interface IIntDPureCut : IDispatch
{
 [propget, id(1), helpstring("property StartTime")]
 HRESULT StartTime([out, retval] DATE *pVal);
 [propput, id(1), helpstring("property StartTime")]
 HRESULT StartTime([in] DATE newVal);

 [propget, id(2), helpstring("property StopTime")]
 HRESULT StopTime([out, retval] DATE *pVal);

 [propget, id(4), helpstring("property SPI")]
 HRESULT SPI([out, retval] long *pVal);
 [propput, id(4), helpstring("property SPI")]
 HRESULT SPI([in] LONG newVal);

 [propget, id(5), helpstring("property RecorderId")]
 HRESULT RecorderId([out, retval] BSTR *pVal);
 [propput, id(5), helpstring("property RecorderId")]
 HRESULT RecorderId([in] BSTR newVal);

 [propget, id(6), helpstring("property ChannelNum")]
 HRESULT ChannelNum([out, retval] short *pVal);
 [propput, id(6), helpstring("property ChannelNum")]
 HRESULT ChannelNum([in] short newVal);

 [propget, id(7), helpstring("property ValidationMessages")]
```



```

 HRESULT ValidationMessages([out, retval] IIntDCutMessageList
**pList);

[propget, id(8), helpstring("property TrailMessages")]
 HRESULT TrailMessages([out, retval] IIntDTrailMessageList **pList);

[propget, id(9), helpstring("property Descriptor")]
 HRESULT Descriptor([out, retval] BSTR *pVal);
[propput, id(9), helpstring("property Descriptor")]
 HRESULT Descriptor([in] BSTR newVal);

[propget, id(10), helpstring("property Edited")]
 HRESULT Edited([out, retval] VARIANT_BOOL *pVal);
[propput, id(10), helpstring("property Edited")]
 HRESULT Edited([in] VARIANT_BOOL newVal);

[propget, id(11), helpstring("property ExternallyValid")]
 HRESULT ExternallyValid([out, retval] VARIANT_BOOL *pVal);

[propget, id(12), helpstring("property InternallyValid")]
 HRESULT InternallyValid([out, retval] VARIANT_BOOL *pVal);

[propget, id(13), helpstring("property ReadyToMerge")]
 HRESULT ReadyToMerge([out, retval] VARIANT_BOOL *pVal);
[propput, id(13), helpstring("property ReadyToMerge")]
 HRESULT ReadyToMerge([in] VARIANT_BOOL newVal);

[propget, id(14), helpstring("property Origin")]
 HRESULT Origin([out, retval] BSTR *pVal);
[propput, id(14), helpstring("property Origin")]
 HRESULT Origin([in] BSTR newVal);

[propget, id(15), helpstring("property UOMCode")]
 HRESULT UOMCode([out, retval] short *pVal);
[propput, id(15), helpstring("property UOMCode")]
 HRESULT UOMCode([in] SHORT newVal);

[propget, id(16), helpstring("property UID")]
 HRESULT UID([out, retval] BSTR *pVal);

[id(50), helpstring("method Load")]
 HRESULT Load(
 [in, defaultvalue(intdLoadValMessages | intdLoadTrails)]IntDLoadFlags
flags,
 [out, retval] IIntDCut **pVal);

[id(51), helpstring("method Remove")]
 HRESULT Remove();

[id(52), helpstring("method Replace")]
 HRESULT Replace([in] IIntDPureCut *Cut, [in] IntDSaveFlags flags);

[id(53), helpstring("method Restore")]
 HRESULT Restore([out, retval] IIntDPureCut **pCut);

[propget, id(54), helpstring("property OriginalDataSource")]
 HRESULT OriginalDataSource([out, retval] BSTR* pVal);

[propput, id(54), helpstring("property OriginalDataSource")]
 HRESULT OriginalDataSource([in] BSTR newVal);
[propget, id(55), helpstring("property Modified")]
 HRESULT Modified([out, retval] VARIANT_BOOL* pVal);
[propget, id(56), helpstring("property TZSTDName")]
 HRESULT TZSTDName([out, retval] BSTR* pVal);
[propput, id(56), helpstring("property TZSTDName")]
 HRESULT TZSTDName([in] BSTR newVal);
[id(57), helpstring("method SaveToXML")]

```

```
HRESULT SaveToXML([in]VARIANT Stream, [out, retval] BSTR* Dest);
};
```

**Property: RecorderId:** Contains the Recorder id.

**Property: ChannelNum:** Contains the Channel number.

**Property: StartTime:** Contains the StartTime of cut.

**Property: StopTime:** Contains the StopTime of cut.

**Property: SPI:** Contains the SPI of cut.

**Property: ValidationMessages:** Returns pointer on an IIntDTrailMessageList interface (look at IIntDTrailMessageList)

**Property: TrailMessages:** Returns pointer on IIntDTrailMessageList interface (look at IIntDTrailMessageList)

**Property: Descriptor:** Contains the Descriptor of cut.

**Property: Edited:** Contains the Edited flag of cut.

**Property: ExternallyValid:** Contains the External validation flag of cut.

**Property: InternallyValid:** Contains the Internal validation flag of cut.

**Property: ReadyToMerge:** Contains the Ready to merge flag of cut.

**Property: Origin:** Contains the Origin flag of cut.

**Property: UOMCode:** Contains the UOM code.

**Property: UID:** Contains the unique identifier of cut for current data source.

**Property: OriginalDataSource:** Contains/allows setting the original datasource of cut(implemented only in WSETProvider).

**Property: Modified:** Contains the Modified flag.

**Property: TZSTDName:** Contains/allows setting time zone standard name.

#### **Method: Load**

Loads interval data and returns a newly created IIntDCut object;

#### **Method: Restore**

Restores a pure cut from archive if the audit is supported by a provider and turned on. This method returns the restored cut from an archived IIntDPureCut object.

#### **Method: Remove**

Removes the cut from datasource.

**Method: Replace**

Replaces a pure cut in the cut list and returns a newly created IIntDPureCut object. If the data source supports the EditTrail functionality, an old cut will be updated according to the EditTrail rules.

**Method: SaveToXML**

Serializes a cut in XML format (see XMLProvider)

Parameter Stream contains the output object. The following VARIANT types are supported

- VT\_EMPTY, XML will be returned as BSTR string (BSTR\* Dest)
- VT\_UNKNOWN, which can contain a value in the form of an IStream, ISequentialStream. XML will be saved by using these interfaces (UTF-8 encoding).

**Enum IntDLoadFlags:** contains type of loading for Load method.

```
typedef [uuid(78785933-336A-4410-A3B8-DA16EC8D57B9), version(1.0)]
enum {
 intdLoadNothing = 0,
 intdLoadValMessages = 1,
 intdLoadTrails = 2,
 intdLoadAsIs = 0,
 intdCheckOverCount = 4,
 intdFillMissing = 8,
 intdCheckUnderCount = 16
} IntDLoadFlags;
```

Value	Description
intdLoadNothing	Do not load any messages
intdLoadValMessages	Load validation messages
intdLoadTrails	Load trail messages
intdLoadAsIs	Load cut as is
intdCheckOverCount	if an interval count more than an expected count returns error
intdFillMissing	Fill missing for DST period if an interval count less than an expected count
intdCheckUnderCount	if an interval count less than an expected count returns error

**Enum IntDSaveFlags:** contains type of saving for Replace and Add methods.

```
typedef [uuid(FA6B7B51-FAE4-4be9-A882-DDFD1264B911), version(1.0)]
enum {
 intdSaveSkipValMessages = 0,
 intdSaveAppendValMessages = 1,
 intdSaveOverValMessages = 2,
 intdSaveSkipTrails = 0,
 intdSaveAppendTrails = 4,
 intdSaveOverTrails = 8
} IntDSaveFlags;
```

Value	Description
intdSaveSkipValMessages	Skip validation messages
intdSaveAppendValMessages	Append validation messages to existing(up to 10)
intdSaveOverValMessages	Overwrite validation messages

IIntDCut Interface

The IIntDCut interface extends the IIntDPureCut interface. Interfaces and methods that appear in bold face are published, others are internal interfaces.

Interface Definition

```
[
 odl,
 uuid(8F340037-5D4E-47c3-99E1-A934D5B90ADE),
 version(1.0),
 dual,
 oleautomation
]
interface IIntDCut : IIntDPureCut {
 [id(0x60020000)]
 HRESULT Scale(
 [in] IntDPeriod period,
 [in] IntDAggregationType aggtype,
 [out, retval] IIntDCut** pRetVal);

 [id(0x60020001)]
 HRESULT ExportXml(
 [in] IntDXmlFormat XmlFormat,
 [in] IntDXmlDataExport DataExport,
 [in] BSTR DateFormat,
 [out, retval] BSTR* pRetVal);

 [id(0x60020002)]
 HRESULT BlockOp(
 [in] IntDBlockOperation blockType,
 [in] IIntDCut* pIntDCut,
 [out, retval] IIntDCut** pRetVal);

 [id(0x60020003)]
 HRESULT BlockOpNA(
 [in] IntDBlockOperation blockType,
 [in] IIntDCut* pIntDCut,
 [out, retval] IIntDCut** pRetVal);

 [id(0x60020004)]
 HRESULT Clone ([out, retval] IIntDCut** pRetVal);

 [id(0x60020005), helpstring("method CopyCut")]
 HRESULT CopyCut([in] BSTR RecorderId, [in] long ChannelId, [in] DATE
 StartTime, [out,
 retval] IIntDCut **pVal);

 [id(0x60020006), helpstring("method IntDCount")]
```

```

HRESULT IntDCount([in] IntDTypeCount type, [out, retval] long *ret);

[id(0x60020007), helpstring("method Value")]
HRESULT Value([in] long index, [out, retval] IIntDCutValue **pVal);

[id(0x60020008), helpstring("method SetValueStatus")]
HRESULT SetValueStatus([in, defaultvalue(intdCOMP_NONE)] IntDCompOperation
op,
 [in] BSTR status,
 [in] BSTR newStatus,
 [in] VARIANT start,
 [in] VARIANT stop,
 [in, optional] VARIANT newValue,
 [out, retval] IIntDCut **pRet);
[id(0x60020009), helpstring("method SetString")]
HRESULT SetString([in] BSTR Status, [out, retval] IIntDCut **pRet);
[id(0x60020010), helpstring("method RemoveSpikes")]
HRESULT RemoveSpikes([in] long SequenceLength,
 [in] double maxValue,
 [in, defaultvalue("K")] BSTR NewStatus);
[id(0x60020011), helpstring("method RemoveDips")]
HRESULT RemoveDips([in] long SequenceLength,
 [in] double minValue,
 [in, defaultvalue("K")] BSTR Status);
[id(0x60020012)]
HRESULT Smooth(
 [in] IntDSmoothType smoothType,
 [in, optional] VARIANT value,
 [out, retval] IIntDCut** pRetVal);

[id(0x60020013), helpstring("method Interpolate")]
HRESULT Interpolate([in] VARIANT Start,
 [in] VARIANT Stop,
 [in, defaultvalue("8")] BSTR QualityCode,
 [in, defaultvalue("J")] BSTR NewStatus);
[id(0x60020014), helpstring("method SpikeTest")]
HRESULT SpikeTest([in] long lPeaks,
 [in] double dPercent,
 [in, defaultvalue("9")] BSTR StatusCode,
 [out, retval] IIntervalList **pRet);
[id(0x60020015), helpstring("method DipTest")]
HRESULT DipTest([in] long lDips,
 [in] double dPercent,
 [in, defaultvalue("9")] BSTR StatusCode,
 [out, retval] IIntervalList **pRet);
[id(0x60020016), helpstring("method Join")]
HRESULT Join([in] IIntDCut *pCut,
 [in] IntDMergeType mrType,
 [in] BOOL bMatchUOM,
 [in] BOOL bValidate,
 [out, retval] IIntDCut **pRet);
[id(0x60020017), helpstring("method Prorate")]
HRESULT Prorate([in, defaultvalue(1.0e100)] double highBound,
 [in, defaultvalue(-1.0e100)] double lowBound,
 [in, defaultvalue("")] BSTR Status,
 [in, optional] VARIANT meterEnergy,
 [in, defaultvalue("")] BSTR newStatus,
 [out, retval] IIntDCut **pVal);
[id(0x60020018)]
HRESULT ScalarOp(
 [in] IntDScaleOperation scaleType,
 [in] VARIANT value,
 [in, defaultvalue("L")] BSTR Status,
 [out, retval] IIntDCut** pRetVal);

[id(0x60020019)]
HRESULT ScalarOpRange(
 [in] IntDScaleOperation scaleType,

```

```

[in] VARIANT value,
[in] DATE StartTime,
[in] DATE StopTime,
[in, defaultvalue("L")] BSTR Status,
[out, retval] IIntDCut** pRetVal);

[id(0x60020020)]
HRESULT Split(
 [in] IntDSplitType splitType,
 [in] DATE SplitTime,
 [out, retval] IIntDCut** pRetVal);

[id(0x60020021)]
HRESULT ReplaceRange(
 [in] double dValue,
 [in] DATE StartTime,
 [in] DATE StopTime,
 [in, defaultvalue("L")] BSTR Status,
 [out, retval] IIntDCut** pRetVal);

[id(0x60020022)]
HRESULT DeleteIntervals(
 [in] DATE StartTime,
 [in] DATE StopTime,
 [out, retval] IIntDCut** pRetVal);

[id(0x60020023)]
HRESULT InsertIntervals(
 [in] double dValue,
 [in] DATE StartTime,
 [in] DATE StopTime,
 [in, defaultvalue("J")] BSTR Status,
 [out, retval] IIntDCut** pRetVal);

[id(0x60020024)]
HRESULT CalcStopTime();

[id(0x60020025)]
HRESULT KnownMeterReading(
 [in] DATE MeterTime,
 [in] double dMeterReading,
 [in, defaultvalue(0)] long lDials,
 [in, defaultvalue(0)] long lDecimals);

[id(0x60020027), helpstring("method ShiftStartTime")]
HRESULT ShiftStartTime(
 [in] IntDShiftType shiftType,
 [in] DATE StartTime);

[id(0x60020028)]
HRESULT InsertIntervalsFromCut(
 [in] DATE StartTime,
 [in] DATE StopTime,
 [in] IIntDCut* pIntDCut,
 [in, defaultvalue("01/01/1970 00:00:00")] DATE StartTime2,
 [in, defaultvalue("**")] BSTR Status,
 [out, retval] IIntDCut** pRetVal);

[id(0x60020029)]
HRESULT ReplaceRangeFromCut(
 [in] DATE StartTime,
 [in] DATE StopTime,
 [in] IIntDCut* pIntDCut,
 [in, defaultvalue("01/01/1970 00:00:00")] DATE StartTime2,
 [in, defaultvalue("**")] BSTR Status,
 [out, retval] IIntDCut** pRetVal);

[id(0x60020030), helpstring("method SetStatuses")]

```

```

HRESULT SetStatuses([in] VARIANT Start, [in] VARIANT Stop, [in] BSTR
Status);

[id(0x60020031), helpstring("method SliderValuesCSV")]
HRESULT SliderValuesCSV([in] LONG Start, [in] LONG Amount, [out,retval]
BSTR* Values);

[id(0x60020032), helpstring("method SliderStatusesCSV")]
HRESULT SliderStatusesCSV([in] LONG Start,
[in] LONG Amount, [out,retval] BSTR* Statuses);
[id(0x60020033), helpstring("method SliderDatesCSV")]
HRESULT SliderDatesCSV([in] LONG Start, [in] LONG Amount,
[in, optional] VARIANT fDateformat, [in, optional] VARIANT fTimeformat,
[in, defaultvalue(" ")] BSTR Separator, [out,retval] BSTR* Dates);

[propget, id(0x60020113), helpstring("property ValuesCSV")]
HRESULT ValuesCSV([out, retval] BSTR *pVal);
[propget, id(0x60020114), helpstring("property StatusesCSV")]
HRESULT StatusesCSV([out, retval] BSTR *pVal);
[propget, id(0x60020115), helpstring("property DatesCSV")]
HRESULT DatesCSV(BSTR DataFormat, [out, retval] BSTR *pVal);

[propget, id(1610744087), helpstring("property DSTParticipant")]
HRESULT DSTParticipant([out, retval] BSTR* pVal);
[propput, id(1610744087), helpstring("property DSTParticipant")]
HRESULT DSTParticipant([in] BSTR newVal);
[propget, id(1610744090), helpstring("property TimeStamp")]
HRESULT TimeStamp([out, retval] VARIANT* pVal);
[propput, id(1610744090), helpstring("property TimeStamp")]
HRESULT TimeStamp([in] VARIANT newVal);
[propget, id(1610744091), helpstring("property UOMName")]
HRESULT UOMName([out, retval] BSTR* pVal);
[propget, id(1610744092), helpstring("property UOMAggregate")]
HRESULT UOMAggregate([out, retval] BSTR* pVal);
[propget, id(1610744093), helpstring("property MaxTime")]
HRESULT MaxTime([out, retval] VARIANT* pVal);
[propget, id(1610744094), helpstring("property Maximum")]
HRESULT Maximum([out, retval] DOUBLE* pVal);
[propget, id(1610744095), helpstring("property PulseMultiplier")]
HRESULT PulseMultiplier([out, retval] DOUBLE* pVal);
[propput, id(1610744095), helpstring("property PulseMultiplier")]
HRESULT PulseMultiplier([in] DOUBLE newVal);
[propget, id(1610744096), helpstring("property FlagE")]
HRESULT FlagE([out, retval] BSTR* pVal);
[propget, id(1610744097), helpstring("property FlagI")]
HRESULT FlagI([out, retval] BSTR* pVal);
[propget, id(1610744098), helpstring("property FlagN")]
HRESULT FlagN([out, retval] BSTR* pVal);
[propget, id(1610744099), helpstring("property FlagO")]
HRESULT FlagO([out, retval] BSTR* pVal);
[propget, id(1610744100), helpstring("property Weight")]
HRESULT Weight([out, retval] DOUBLE* pVal);
[propput, id(1610744100), helpstring("property Weight")]
HRESULT Weight([in] DOUBLE newVal);
[propget, id(1610744101), helpstring("property Population")]
HRESULT Population([out, retval] DOUBLE* pVal);
[propput, id(1610744101), helpstring("property Population")]
HRESULT Population([in] DOUBLE newVal);
[propget, id(1610744102), helpstring("property AbsMaxTime")]
HRESULT AbsMaxTime([out, retval] VARIANT* pVal);
[propget, id(1610744103), helpstring("property AbsMaximum")]
HRESULT AbsMaximum([out, retval] DOUBLE* pVal);
[propget, id(1610744104), helpstring("property KWMax")]
HRESULT KWMax([out, retval] DOUBLE* pVal);
[propget, id(1610744105), helpstring("property Minimum")]
HRESULT Minimum([out, retval] DOUBLE* pVal);
[propget, id(1610744106), helpstring("property MinTime")]
HRESULT MinTime([out, retval] VARIANT* pVal);

```

```

 [propget, id(1610744107), helpstring("property MinimumNZ")]
 HRESULT MinimumNZ([out, retval] DOUBLE* pVal);
 [propget, id(1610744108), helpstring("property CountNZ")]
 HRESULT CountNZ([out, retval] LONG* pVal);
 [propget, id(1610744109), helpstring("property CountNon9Val")]
 HRESULT CountNon9Val([out, retval] LONG* pVal);
 [propget, id(1610744110), helpstring("property Total")]
 HRESULT Total([out, retval] DOUBLE* pVal);
 [propget, id(1610744111), helpstring("property Energy")]
 HRESULT Energy([out, retval] DOUBLE* pVal);
 [propget, id(1610744112), helpstring("property Average")]
 HRESULT Average([out, retval] DOUBLE* pVal);
 [propget, id(1610744113), helpstring("property AverageNZ")]
 HRESULT AverageNZ([out, retval] DOUBLE* pVal);
 [propget, id(1610744114), helpstring("property MaximumN")]
 HRESULT MaximumN([in, defaultvalue(1)] LONG Index, [out, retval] DOUBLE*
pVal);
 [propget, id(1610744115), helpstring("property MaxTimeN")]
 HRESULT MaxTimeN([in, defaultvalue(1)] LONG Index, [out, retval] VARIANT*
pVal);
 [propget, id(1610744126), helpstring("property IPH")]
 HRESULT IPH([out, retval] LONG* pVal);
 [propget, id(1610744129), helpstring("property DeleteFlag")]
 HRESULT DeleteFlag([out, retval] VARIANT_BOOL* pVal);
 [propget, id(1610744129), helpstring("property DeleteFlag")]
 HRESULT DeleteFlag([in] VARIANT_BOOL newVal);
 [propget, id(1610744130), helpstring("property EditedRulsLang")]
 HRESULT EditedRulsLang([out, retval] VARIANT_BOOL* pVal);
 [propget, id(1610744131), helpstring("property LoadFactor")]
 HRESULT LoadFactor([out, retval] DOUBLE* pVal);
 [propget, id(1610744132), helpstring("property MeterStartReading")]
 HRESULT MeterStartReading([out, retval] DOUBLE* pVal);
 [propget, id(1610744132), helpstring("property MeterStartReading")]
 HRESULT MeterStartReading([in] DOUBLE newVal);
 [propget, id(1610744133), helpstring("property MeterStopReading")]
 HRESULT MeterStopReading([out, retval] DOUBLE* pVal);
 [propget, id(1610744133), helpstring("property MeterStopReading")]
 HRESULT MeterStopReading([in] DOUBLE newVal);
 [propget, id(1610744134), helpstring("property MeterOffset")]
 HRESULT MeterOffset([out, retval] DOUBLE* pVal);
 [propget, id(1610744134), helpstring("property MeterOffset")]
 HRESULT MeterOffset([in] DOUBLE newVal);
 [propget, id(1610744135), helpstring("property MeterMultiplier")]
 HRESULT MeterMultiplier([out, retval] DOUBLE* pVal);
 [propget, id(1610744135), helpstring("property MeterMultiplier")]
 HRESULT MeterMultiplier([in] DOUBLE newVal);
 [propget, id(1610744136), helpstring("property Timezone")]
 HRESULT Timezone([out, retval] SHORT* pVal);
 [propget, id(1610744136), helpstring("property Timezone")]
 HRESULT Timezone([in] SHORT newVal);
 [propget, id(1610744137), helpstring("property UOMTotalize")]
 HRESULT UOMTotalize([out, retval] BSTR* pVal);
 [propget, id(1610744138), helpstring("property UOMRateQuantity")]
 HRESULT UOMRateQuantity([out, retval] BSTR* pVal);
 [propget, id(1610744139), helpstring("property UOMRelatedCode")]
 HRESULT UOMRelatedCode([out, retval] SHORT* pVal);
 [propget, id(1610744140), helpstring("property PulseOffset")]
 HRESULT PulseOffset([out, retval] DOUBLE* pVal);
 [propget, id(1610744140), helpstring("property PulseOffset")]
 HRESULT PulseOffset([in] DOUBLE newVal);
 };

```

Published Properties are self evident and are properties of the Cut Object. Some may be set, some are read only. See **Appendix B: Oracle Utilities Enhanced Input/Output Interval Data Format** in the *Data Manager User's Guide* for details on individual properties.



**Published Method: Scale**

This method is encapsulated within the IIntDCut interface. It creates and returns the address of a new IIntDCut object, where each interval has the SPI of the specified period. It aggregates the intervals of the current cut, based on the aggregation type (either add or average).

The first argument sent to the method is the period. The second argument sent to the method is the aggrtype. A newly created IIntDCut object conforming to the request is returned to the calling procedure.

**Published Method: Split**

This method splits a cut in two at the time you specify.

```
HRESULT Split(
 [in] IntDSplitType splitType,
 [in] DATE SplitTime,
 [out, retval] IIntDCut** pRetVal);
```

This method accepts 2 arguments and returns the first or second cut based on the splitType. A newly created IIntDCut object conforming to the request is returned to the calling procedure.

- **splitType.** This parameter specifies whether the first or second cut should be returned.
- **SplitTime:** A Date Value representing the time that the original cut will be split. The time entered must be between the start and stop times of the original cut. This time becomes the start time of the second cut created by the split. A time of 1 second before this is calculated as the stop time of the first cut.
- **Enum IntDSplitType:** contains the split operations that can be performed on cuts.

```
typedef [uuid(e576dc30-a022-41ce-86aa-50a9e37bb01d), version(1.0)]
enum {
 intdSPLIT_FIRST = 0,
 intdSPLIT_SECOND = 1
} IntDSplitType;
```

Value	Description
intdSPLIT_FIRST	Return first split cut.
intdSPLIT_SECOND	Return second split cut.

**Published Method: CalcStopTime**

This method calculates the stop time of the cut based on the start time of the cut, the number of intervals, the DST\_Participant flag, and the seconds-per-interval.

```
HRESULT CalcStopTime();
```

The re-computed stop-time is stored in the cut. This method is used to fix an incorrectly transcribed stop-time, or to force the cut's expected intervals to match the recorded intervals.

## IIntDCutValue Interface

The IIntdCutValue interface provides access to values and status codes in the Cut. It is created via the Value method of the IIntDCut object.

**Interface Definition**

```
[
 odl,
 uuid(9ED54F84-A89D-4FCD-A854-44251E925F09),
 version(1.0),
 dual,
 oleautomation
]
interface IIntdCutValue : IDispatch {
[id(0x60020000), propget]
 HRESULT Value([out, retval] double* pRetVal);
[id(0x60020000), propput]
 HRESULT Value([in] double pRetVal);
[id(0x60020001), propget]
 HRESULT Status([out, retval] BYTE* pRetVal);
[id(0x60020001), propput]
 HRESULT Status([in] BYTE pRetVal);
[id(0x60020002), propget]
 HRESULT DateTime([out, retval] DATE* pRetVal);
[id(0x60020002), propput]
 HRESULT DateTime([in] DATE* pRetVal);
};
```

**Property: Value:** Allows users to set and get of the values or intervals.

**Property: Status:** Allows users to set and get the status codes.

## IIntDProvider Interface

The IIntDProvider interface allows a connection to a data source provider.

```
[
 odl,
 uuid(02F59F2D-B197-4b77-BB55-C17742C5379F),
 version(1.0),
 dual,
 oleautomation
]
interface IIntDProvider : IDispatch {
 [id(0x60020000)]
 HRESULT Connect([in] BSTR ConnectString);
};
```

### Method: Connect

The IIntDDataSource object uses this method to connect a provider to a particular data source.

# Interval Data Source Providers

## RDBProvider

The RDBProvider provides ability to use a relational database as an interval datasource. The RDBProvider is the COM component that implements the interface of interval datasource provider.

**ProgID:** LSIntD.RDBProvider

**Provider-specific connection string:** RDBProvider accepts following XML structure as the connection string:

```
<PROVIDER PROGID="LSIntD.RDBProvider">
 <DATASOURCE>
 . . . LODESTAR's DATASOURCE structure
 </DATASOURCE>
 [<CUTHEADERTABLE>string</CUTHEADERTABLE>]
</PROVIDER>
```

This XML structure contains the following XML elements:

- **DATASOURCE:** (Required) Standard DATASOURCE structure. Default value is N/A
- **CUTHEADERTABLE:**(Optional) Contains the name of a table in the data source which contains headers of cuts. Default value is LSCHANNELCUTHEADER

## BTEProvider

This provider will allow for access to interval data in a Pervasive database. (filename.bte)

**ProgID:** LSIntD.BTEProvider

**Provider-specific connection string:** The BTEProvider accepts following XML structure as the Provider string:

```
<PROVIDER PROGID="LSIntD.BTEProvider"
 PATH="String"
</PROVIDER>
```

This XML structure contains the following XML elements:

- **PATH:** (Required) Fully qualified path the BTE file set. Default value is N/A.

## DATASOURCE Definition

The DTD for a valid DATASOURCE XML structure is shown below.

```
<!DOCTYPE DATASOURCE
[
<!ELEMENT DATASOURCE (NAME, CONNECTSTRING?, USERID?, PASSWORD?,
 QUALIFIER?)>
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT CONNECTSTRING (#PCDATA)>
<!ELEMENT USERID (#PCDATA)>
<!ELEMENT PASSWORD (#PCDATA)>
<!ELEMENT QUALIFIER (#PCDATA)>
]>
```

The first time that a DataSource object is initialized with a unique data source name, a static structure will be cached to store information related to the data source. The first data source that is cached will become the default data source. Any subsequent initialization of a DataSource object with a previously used data source name will create an object representing the previously cached structure.

**Element Descriptions:** The only required argument for DataSource initialization is the data source name (except for the default constructor). The ConnectString argument must be provided prior to actually creating a Connection object. The first UserID, Password, or Qualifier argument provided for each unique data source will become the default for that data source.

## Examples

This example DATASOURCE XML string is an example of how the XML should look if connecting to a BTE file using many defaulted settings.

```
<INTDATASOURCE>
<PROVIDER
 PROGID='LSIntD.BTEProvider'
 PATH='\"d:\btefiles\mybtefile.bte\"'>
</PROVIDER>
<SESSION/>
</INTDATASOURCE>
```

This example DATASOURCE XML string is an example of how the XML should look if connecting to a RDB Interval data store.

```
<INTDATASOURCE>
 <PROVIDER PROGID="LSIntD.RDBProvider">
 <DATASOURCE>
 <NAME>INTDTEST</NAME>
 <CONNECTSTRING>DSN=MYODBC;UID=MYUSERID;PWD=MYPASSWORD;</CONNECTSTRING>
 <QUALIFIER>MYQUALIFIER</QUALIFIER>
 </DATASOURCE>
 <CUTHEADERTABLE>LSCHANNELCUTHEADER</CUTHEADERTABLE>
 </PROVIDER>
 <SESSION>
 <INTD_NEW_JOIN INTDMERGESELECT="RECENT" CHECKVALID="Y"/>
 <CHECK_UOM_MATCHING CHECK_COMPATIBILITY="Y"/>
 </SESSION>
</INTDATASOURCE>
```

## Sample Application using LSINTD

This section contains a sample application that uses the LSINTD component.

```
Private Sub cmdExtractAll_Click()
' This subroutine is not very optimized.
' You may not want to load node details unless user clicks a node.
 Dim FileName As String 'Filename of the BTE file
 Dim objectIntd As LSINTDLib.IntDDDataSource 'the datasource object
 Dim objRecorder As LSINTDLib.IIntDRecorder '
 Dim objChannel As LSINTDLib.IIntDChannel
 Dim objCuts As LSINTDLib.IIntDCutList
 Dim objPureCut As LSINTDLib.IIntDPureCut
 Dim ObjCut As LSINTDLib.IIntDCut
 Dim objIntervalValue As LSINTDLib.IIntDCutValue
 Dim iIntervalCount As Integer
 Dim itemp As Integer
 Dim iIndex As Long
 Dim dblIntervalValue As Double
 Dim strIntervalStatusValue As String
 Dim strOutput As String
 Dim iNumProcessed As Long
 Dim strReference As String
 Dim strChanReference As String

 On Error GoTo ErrorHandler
 Me.tvwOutput.Visible = True
 Me.lstOutput.Visible = False
 Me.txtOutput.Visible = False
 'prepare
 FileName = txtFileName.Text
 If FileName = "" Then Exit Sub
 cmdExit.Enabled = False
 glbCancelLoad = False
 'strOutput = ""
 Me.tvwOutput.Nodes.Clear
 Me.txtTotalLoaded.Text = "0"
 Me.pbLoadProgress.Visible = True
 Me.lblProgress.Visible = True

 'instantiate the datasource object and create the session
 'Set ObjectIntd = CreateObject("LSIntD.IntdDataSource")
 Set objectIntd = New LSINTDLib.IntDDDataSource
 objectIntd.Connect "<INTDATASOURCE><PROVIDER PROGID='LSIntD.BTEProvider'
 PATH='" + _ FileName + "'><USERID/></PROVIDER><SESSION/></INTDATASOURCE>"

 'Set the progress bar
 Me.pbLoadProgress.Max = objectIntd.Recorders.Count
 Me.txtTotalRecorders = CStr(objectIntd.Recorders.Count)
 iNumProcessed = 1

 'loop through the collection of recorders in the datasource object
 For Each objRecorder In objectIntd.Recorders
 DoEvents 'Allows for cancel button to interrupt
 If glbCancelLoad = True Then
 Exit For
 End If

 Me.pbLoadProgress.Value = iNumProcessed
 Me.txtTotalLoaded.Text = CStr(iNumProcessed)

 'Get properties and display to user for recorder level
 'Add the recorder ID as a node
 Me.tvwOutput.Nodes.Add , , objRecorder.RecorderId, objRecorder.RecorderId
 & " [" & objRecorder.Channels.Count & " channels]"

 'Loop through the collection of channels in the recorder object
 For Each objChannel In objRecorder.Channels
```

```

'Add a channel node
strChanReference = objRecorder.RecorderId & CStr(objChannel.ChannelNum)
Me.tvwOutput.Nodes.Add objRecorder.RecorderId, tvwChild,
strChanReference, objChannel.ChannelNum

'Cuts is a cutlist object that contains cuts from the channel
'Get cuts from the channel and set to Cuts object
Set objCuts = objChannel.Cuts

'We have a list of all LodeStar cuts in the channel object
'Loop through the cuts and display info about them (properties)
For Each objPureCut In objCuts

 'strOutput = strOutput + vbTab & vbTab & "Cut : " &
objPureCut.StartTime & " - " & objPureCut.StopTime & vbCrLf
 'Create the cut Status object
 On Error Resume Next
 Set ObjCut = objPureCut.Load() 'use debug to view properites
 If Err.Number <> 0 Then
 GoTo ErrorLabel
 End If
 On Error GoTo ErrorHandler

 'Add a cut node for a pure cut
 strReference = objRecorder.RecorderId & CStr(objChannel.ChannelNum) &
CStr(objPureCut.StartTime)
 Me.tvwOutput.Nodes.Add strChanReference, tvwChild, strReference,
objPureCut.StartTime

 'Add Cut information:
 strOutput = "StartTime: " & CStr(ObjCut.StartTime)
 Me.tvwOutput.Nodes.Add strReference, tvwChild, , strOutput

 strOutput = "StopTime: " & CStr(ObjCut.StopTime)
 Me.tvwOutput.Nodes.Add strReference, tvwChild, , strOutput

 strOutput = "StopTime: " & CStr(ObjCut.StopTime)
 Me.tvwOutput.Nodes.Add strReference, tvwChild, , strOutput

 strOutput = "Meter Start: " & CStr(ObjCut.MeterStartReading)
 Me.tvwOutput.Nodes.Add strReference, tvwChild, , strOutput

 strOutput = "Meter Stop: " & CStr(ObjCut.MeterStopReading)
 Me.tvwOutput.Nodes.Add strReference, tvwChild, , strOutput

 strOutput = "Meter Mult: " & CStr(ObjCut.MeterMultiplier)
 Me.tvwOutput.Nodes.Add strReference, tvwChild, , strOutput

 strOutput = "Pulse Mult: " & CStr(ObjCut.PulseMultiplier)
 Me.tvwOutput.Nodes.Add strReference, tvwChild, , strOutput

 strOutput = "Pulse Offset: " & CStr(ObjCut.PulseOffset)
 Me.tvwOutput.Nodes.Add strReference, tvwChild, , strOutput

 strOutput = "Interval Total: " & CStr(ObjCut.Total)
 Me.tvwOutput.Nodes.Add strReference, tvwChild, , strOutput

 strOutput = "Population: " & CStr(ObjCut.Population)
 Me.tvwOutput.Nodes.Add strReference, tvwChild, , strOutput

 strOutput = "Weight: " & CStr(ObjCut.Weight)
 Me.tvwOutput.Nodes.Add strReference, tvwChild, , strOutput

 strOutput = "SPI: " & CStr(ObjCut.SPI)
 Me.tvwOutput.Nodes.Add strReference, tvwChild, , strOutput

 strOutput = "Interval Count: " & CStr(ObjCut.IntervalCount)

```

```
Me.tvwOutput.Nodes.Add strReference, tvwChild, , strOutput

strOutput = "UOM: " & CStr(ObjCut.UOMCode)
Me.tvwOutput.Nodes.Add strReference, tvwChild, , strOutput

strOutput = "TsWoGMT: " & CStr(ObjCut.Timezone)
Me.tvwOutput.Nodes.Add strReference, tvwChild, , strOutput

strOutput = "Descriptor: " & CStr(ObjCut.Descriptor)
Me.tvwOutput.Nodes.Add strReference, tvwChild, , strOutput

strOutput = "Validation Messages " & "[" &
ObjCut.ValidationMessages.Count & "]"
Dim strVmsgParent As String
strVmsgParent = objRecorder.RecorderId & CStr(objChannel.ChannelNum)
& CStr(objPureCut.StartTime) & "V"
Me.tvwOutput.Nodes.Add strReference, tvwChild, strVmsgParent,
strOutput
'Add vmsg here as child key =
If ObjCut.ValidationMessages.Count >= 1 Then

 iIndex = 1
 Do While iIndex <= ObjCut.ValidationMessages.Count
 'DoEvents
 strOutput = "V Message #" & CStr(iIndex) & " = " &
ObjCut.ValidationMessages(iIndex - 1).Text
 Me.tvwOutput.Nodes.Add strVmsgParent, tvwChild, , strOutput
 iIndex = iIndex + 1
 Loop
End If

strOutput = "TZSN: " & CStr(ObjCut.TZSTDName)
Me.tvwOutput.Nodes.Add strReference, tvwChild, , strOutput

strOutput = "Edited: " & CStr(ObjCut.Edited)
Me.tvwOutput.Nodes.Add strReference, tvwChild, , strOutput

strOutput = "Edited: " & CStr(ObjCut.Edited)
Me.tvwOutput.Nodes.Add strReference, tvwChild, , strOutput

'Prepare for values and status codes for the cut.
iIntervalCount = ObjCut.IntervalCount
itemp = 1
'This loop gets the actual intervals and status codes of the pure cut.
Do While itemp <= iIntervalCount
 Set objIntervalValue = ObjCut.Value(itemp)
 dblIntervalValue = objIntervalValue.Value 'not actually doing
anything with the values and statuses
 strIntervalStatusValue = objIntervalValue.Status
 itemp = itemp + 1
Loop
ErrorLabel:
 Set ObjCut = Nothing
Next
Set objCuts = Nothing
Next
iNumProcessed = iNumProcessed + 1
Next

'txtOutput.Text = strOutput
'Release the objects
Me.MousePointer = vbNormal
Set objectIntd = Nothing
Set objRecorder = Nothing
Set objChannel = Nothing
Set objCuts = Nothing
Set ObjCut = Nothing
```



```

 Set objPureCut = Nothing
 Set objIntervalValue = Nothing
 Me.pbLoadProgress.Visible = False
 Me.lblProgress.Visible = False
 cmdExit.Enabled = True
 Exit Sub
 ErrorHandler:

 Me.pbLoadProgress.Visible = False
 Me.lblProgress.Visible = False
 cmdExit.Enabled = True
 Me.MousePointer = vbNormal
 MsgBox "Error No.: " & Err.Number & vbCrLf & "Description: " & Err.Description

 End Sub

Private Sub cmdExtractCut_Click()
 Dim FileName As String 'Filename of the BTE file
 Dim objectIntd As LSINTDLib.IntDDataSource 'the datasource object
 Dim objPureCut As LSINTDLib.IIntDPureCut
 Dim ObjCut As LSINTDLib.IIntDCut
 Dim objIntervalValue As LSINTDLib.IIntDCutValue
 Dim iIntervalCount As Integer
 Dim iIndex As Integer
 Dim itemp As Integer
 Dim dblIntervalValue As Double
 Dim strIntervalStatusValue As String
 Dim strOutput As String
 Dim datStartDate As Date
 Dim DateSupplied As Boolean
 Dim ReturnCode As Long

 Me.tvwOutput.Visible = False
 Me.lstOutput.Visible = False
 Me.txtOutput.Visible = True
 On Error GoTo ErrorHandler
 'Set up...
 Me.txtTotalRecorders = ""
 Me.txtTotalLoaded = ""
 glbCancelLoad = False
 'Me.txtOutput.Text = ""
 'Me.MousePointer = vbHourglass
 strOutput = ""
 FileName = txtFileName.Text
 If FileName = "" Then Exit Sub

 'instantiate the datasource object and create the session
 'Set ObjectIntd = CreateObject("LSIntD.IntdDataSource")
 Set objectIntd = New LSINTDLib.IntDDataSource
 objectIntd.Connect "<INTDATASOURCE><PROVIDER PROGID='LSIntD.BTEProvider'
 PATH='" + FileName + "'><USERID/></PROVIDER><SESSION/></INTDATASOURCE>"
 DateSupplied = True
 If Not IsDate(Me.txtStartTime) Then
 DateSupplied = False
 MsgBox ("Invalid date.")
 Exit Sub
 Else
 datStartDate = CDate(Me.txtStartTime)
 'Get the Pure Cut
 Set objPureCut = objectIntd.GetCutByKey(UCase(Me.txtRecorderID),
 CLng(Me.txtChannel), datStartDate)
 End If
 'Example of Loading a cut over a date range
 'Set ObjCut = objectIntd.LoadDates(Recorder, Channel, StartDate, StopDate)

 Set ObjCut = objPureCut.Load()

 'Format the string of cut attributes for display

```

```
ReturnCode = FormatOutput(ObjCut, strOutput)
Me.txtOutput.Text = strOutput
'Prepare for values and status codes for the cut.

'This loop gets the actual intervals and status codes of the pure cut.
'These are only for viewing inb the debugger.
'Not outputted to the results control.
'Illustrates how to get individual intervals and statuses
'Uncomment the following Lines if required
'iIntervalCount = ObjCut.IntervalCount
'itemp = 1
'Do While itemp <= iIntervalCount
' Set objIntervalValue = ObjCut.Value(itemp)
' dblIntervalValue = objIntervalValue.Value 'not actually doing anything
with the values and statuses
' strIntervalStatusValue = objIntervalValue.Status
' itemp = itemp + 1
'Loop

'Release the objects
Me.MousePointer = vbNormal
Set objectIntd = Nothing
Set objPureCut = Nothing
Set objIntervalValue = Nothing
Exit Sub
ErrorHandler:
Me.MousePointer = vbNormal
MsgBox "Error No.: " & Err.Number & vbCrLf & "Description: " & Err.Description
End Sub

Function FormatOutput(ObjCut As LSINTDLib.IIntDCut, strOutput As String) As
Long

Dim iIndex As Long

strOutput = strOutput & "RecID: " & vbTab & vbTab & ObjCut.RecorderId & vbCrLf
strOutput = strOutput & "Channel: " & vbTab & vbTab & ObjCut.ChannelNum &
vbCrLf
strOutput = strOutput & "Start: " & vbTab & vbTab & ObjCut.StartTime & vbCrLf
strOutput = strOutput & "Stop: " & vbTab & vbTab & ObjCut.StopTime & vbCrLf
strOutput = strOutput & "Met Start: " & vbTab & ObjCut.MeterStartReading &
vbCrLf
strOutput = strOutput & "Met Stop: " & vbTab & ObjCut.MeterStopReading &
vbCrLf
strOutput = strOutput & "Met Mult: " & vbTab & vbTab & ObjCut.MeterMultiplier
& vbCrLf
strOutput = strOutput & "Int Total: " & vbTab & vbTab & ObjCut.Total & vbCrLf
strOutput = strOutput & "SPI: " & vbTab & vbTab & ObjCut.SPI & vbCrLf
strOutput = strOutput & "UOM: " & vbTab & vbTab & ObjCut.UOMCode & vbCrLf
strOutput = strOutput & "Desc: " & vbTab & vbTab & ObjCut.Descriptor & vbCrLf
strOutput = strOutput & "Num ValMsgs: " & vbTab &
ObjCut.ValidationMessages.Count & vbCrLf

If ObjCut.ValidationMessages.Count > 0 Then
iIndex = 1
Do While iIndex <= ObjCut.ValidationMessages.Count
strOutput = strOutput & "ValMsg" & iIndex & ": " & _
vbTab & ObjCut.ValidationMessages.Item(iIndex - 1).Text &
vbCrLf
iIndex = iIndex + 1
Loop
End If

strOutput = strOutput & "TZSN: " & vbTab & vbTab & ObjCut.TZSTDName & vbCrLf
strOutput = strOutput & "Edited: " & vbTab & vbTab & ObjCut.Edited & vbCrLf
strOutput = strOutput & "Int Val: " & vbTab & vbTab & ObjCut.InternallyValid
& vbCrLf
```

```
strOutput = strOutput & "Ext Val: " & vbTab & vbTab & ObjCut.ExternallyValid
& vbCrLf
strOutput = strOutput & "Merge: " & vbTab & vbTab & ObjCut.ReadyToMerge &
vbCrLf
strOutput = strOutput & "Archv: " & vbTab & vbTab & ObjCut.DeleteFlag & vbCrLf
strOutput = strOutput & "Origin: " & vbTab & vbTab & ObjCut.Origin & vbCrLf
strOutput = strOutput & "DST Flag: " & vbTab & ObjCut.DSTParticipant & vbCrLf
strOutput = strOutput & "Edited By RS: " & vbTab & ObjCut.EditedRulsLang &
vbCrLf

strOutput = strOutput & vbCrLf
strOutput = strOutput & "STATISTICS:" & vbCrLf
strOutput = strOutput & "Energy: " & vbTab & vbTab & ObjCut.Energy & vbCrLf

strOutput = strOutput & "Avg Int: " & vbTab & vbTab & ObjCut.Average & vbCrLf
strOutput = strOutput & "Avg NZ: " & vbTab & vbTab & ObjCut.AverageNZ & vbCrLf
strOutput = strOutput & "Peak: " & vbTab & vbTab & ObjCut.AbsMaxTime & vbCrLf
FormatOutput = 0
Exit Function
End Function
```

## INTDVB Compatibility

This section describes corresponding interfaces and methods in the LSINTD component to the functions in the INTDVB component.

To begin using the LSINTD.DLL in VB, the programmer must add the Reference to the VB Project. This will allow for the use of the Interface in the MS IDE.

### INTDVB - LSINTD Mapping

INTDVB Function Name	LSINTD Interface, Method
IntDataOpenSession	<pre>Dim objectIntd As LSINTDLib.IntDDDataSource 'the datasource object  Set objectIntd = New LSINTDLib.IntDDDataSource objectIntd.Connect "&lt;INTDATASOURCE&gt;&lt;PROVIDER PROGID='LSIntD.BTEProvider' PATH='" + FileName + "'&gt;&lt;USERID/&gt;&lt;/ PROVIDER&gt;&lt;SESSION/&gt;&lt;/ INTDATASOURCE&gt;"</pre>
IntDataGetError	<p>This function is not necessary. Errors may be obtained from the Err object.</p> <p>Use an Error Handler.</p>
IntDataCloseSession	<p>LSINTD method not needed. Set the Object to Nothing.</p> <p>Set objectIntd = Nothing</p>
IntDataGetCutHeaders	<pre>Dim objRecorder As LSINTDLib.IIntDRecorder ' Dim objChannel As LSINTDLib.IIntDChannel Dim objCuts As LSINTDLib.IIntDCutList Dim ObjCut As LSINTDLib.IIntDCut  Dim objIntervalValue As LSINTDLib.IIntDCutValue  For Each objRecorder In objectIntd.Recorders For Each objChannel In objRecorder.Channels Set objCuts = objChannel.Cuts  'We have a list of all LodeStar cuts in the channel object  'Loop through the cuts and display info about them (properties)</pre>

INTDVB Function Name	LSINTD Interface, Method
IntDataGetCutHeaderValue	<p>For Each objPureCut In objCuts</p> <p>'Load the Cut's info and Properties</p> <p>Set ObjCut = objPureCut.Load()</p> <p>strReference = objRecorder.RecorderId &amp; _  CStr(objChannel.ChannelNum) &amp; _  CStr(objPureCut.StartTime)</p>
IntDataReleaseCutHeaderArray	Not Needed. Set the Objects to Nothing.
IntDataLoadCut	<p>Dim objectIntd As LSINTDLib.IntDDDataSource</p> <p>'the datasource object</p> <p>Dim objPureCut As LSINTDLib.IIntDPureCut</p> <p>Dim ObjCut As LSINTDLib.IIntDCut</p> <p>Dim objIntervalValue As  LSINTDLib.IIntDCutValue</p> <p>Set objectIntd = New LSINTDLib.IntDDDataSource</p> <p>objectIntd.Connect</p> <p>"&lt;INTDATASOURCE&gt;&lt;PROVIDER  PROGID='LSIntD.BTEProvider' PATH="" +  FileName + ""&gt;&lt;USERID/&gt;&lt;/  PROVIDER&gt;&lt;SESSION/&gt;&lt;/  INTDATASOURCE&gt;"</p> <p>Set objPureCut =  objectIntd.GetCutByKey(UCase(Me.txtRecorderID)  , _  CLng(Me.txtChannel), datStartDate)</p> <p>Set ObjCut = objPureCut.Load()</p> <p>"This loop gets the actual intervals and status codes  of the pure cut.</p> <p>'Illustrates how to get individual intervals and  statuses</p> <p>iIntervalCount = ObjCut.IntervalCount</p> <p>itemp = 1</p> <p>Do While itemp &lt;= iIntervalCount</p> <p>Set objIntervalValue = ObjCut.Value(itemp)</p> <p>dblIntervalValue = objIntervalValue.Value</p> <p>strIntervalStatusValue = objIntervalValue.Status</p> <p>itemp = itemp + 1</p> <p>Loop</p>
IntDataLoadCutEx	<p>Set ObjCut = objectIntd.LoadDates(Recorder,  Channel, StartDate, _ StopDate)</p>

INTDVB Function Name	LSINTD Interface, Method
IntDataSaveCut	objPureCut = objectIntd.Cuts.Add(objPureCut, 0) objPureCut = ObjCut.Replace(objPureCut, 0)
IntDataReleaseCut	Set the Object to Nothing
IntDataGetCutIntervals	Not needed. See IntDataLoadCut
IntDataGetCutValues	Not needed. See IntDataLoadCut
IntDataSubset	Perform two calls to the Split Method. ObjCut1 = ObjCut.Split(0, ObjCut) ObjCut2 = ObjCut.Split(1, ObjCut)
IntDataDelete	This was an undocumented interface in INTDVB. Use the Remove Method of LSINTD. ObjCut.Remove

# Part Four

---

## Appendices

Part Four includes appendices that describe import and export file formats used by the Oracle Utilities Energy Information Platform, and contains the following chapters:

- **Appendix A: Oracle Utilities Data Repository Database Schema**
- **Appendix B: Oracle Utilities Data Repository Schema Import File Format**
- **Appendix C: Oracle Utilities Enhanced Input/Output Interval Data Format**
- **Appendix D: Oracle Utilities Standard XML Interval Data Format**
- **Appendix E: Oracle Utilities Comma Separated Values (CSV) Interval Data Format**

---



# Appendix A

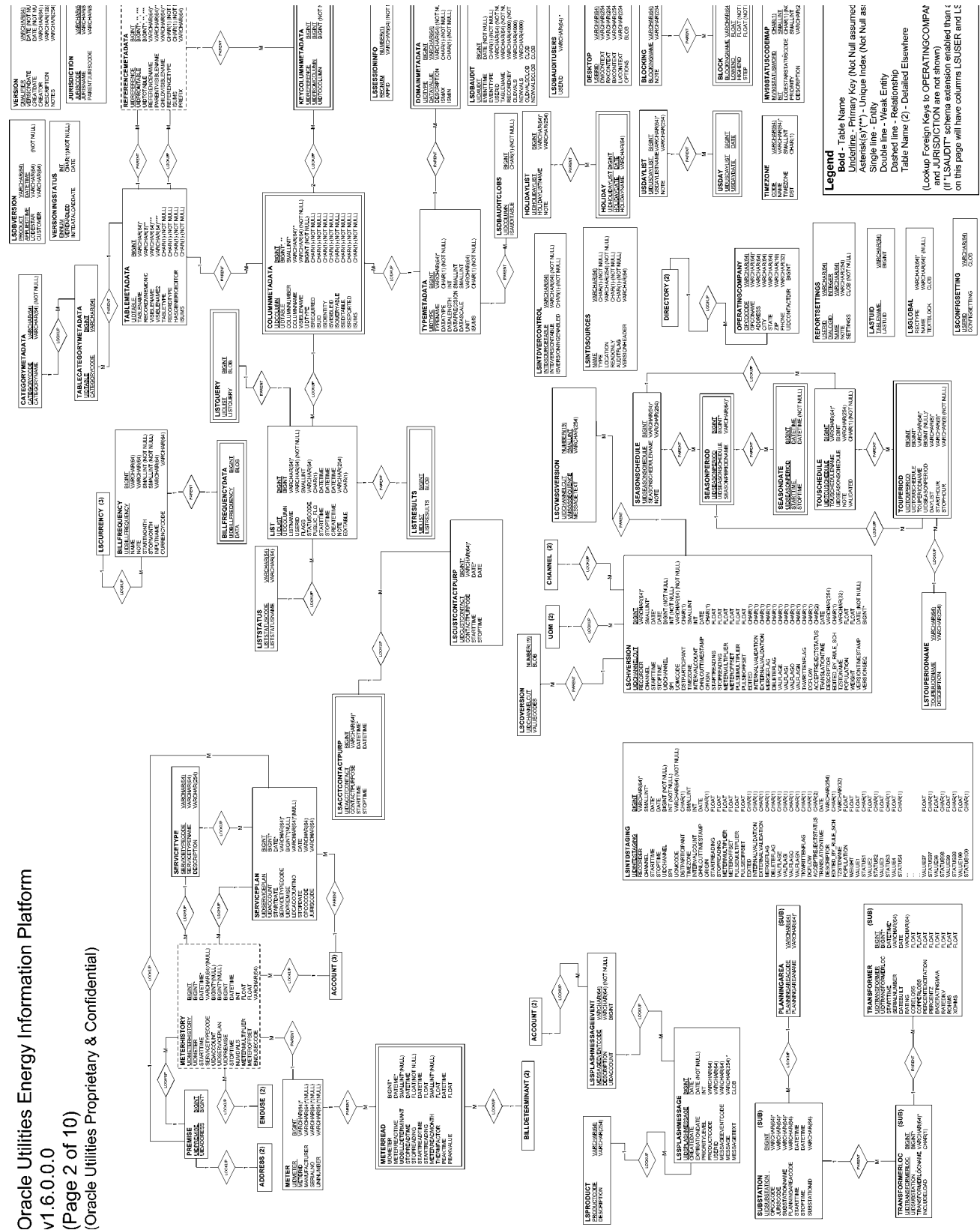
---

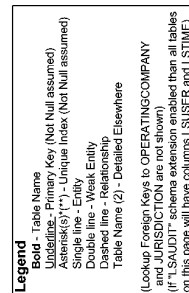
## Oracle Utilities Data Repository Database Schema

This appendix includes a diagram of the Oracle Utilities Data Repository database schema (v1.6.1.0.0) that provides details regarding the table and columns in the schema, as well as the relationships between these tables in the Oracle Utilities Data Repository. This information is very useful when writing Rules Language statements or constructing database queries. This includes:

- **Energy Information Platform Schema p. 1**
- **Energy Information Platform Schema p. 2**
- **Energy Information Platform Schema - Security**
- **Energy Information Platform Schema - Reporting**
- **Energy Information Platform Schema - Messaging**
- **Energy Information Platform Schema - Work Queues**
- **Energy Information Platform Schema - Interval Data Manager**
- **Energy Information Platform Schema - Adapter**
- **Energy Information Platform Schema - Service Points and Market Participants**
- **Energy Information Platform Schema - Web Services**

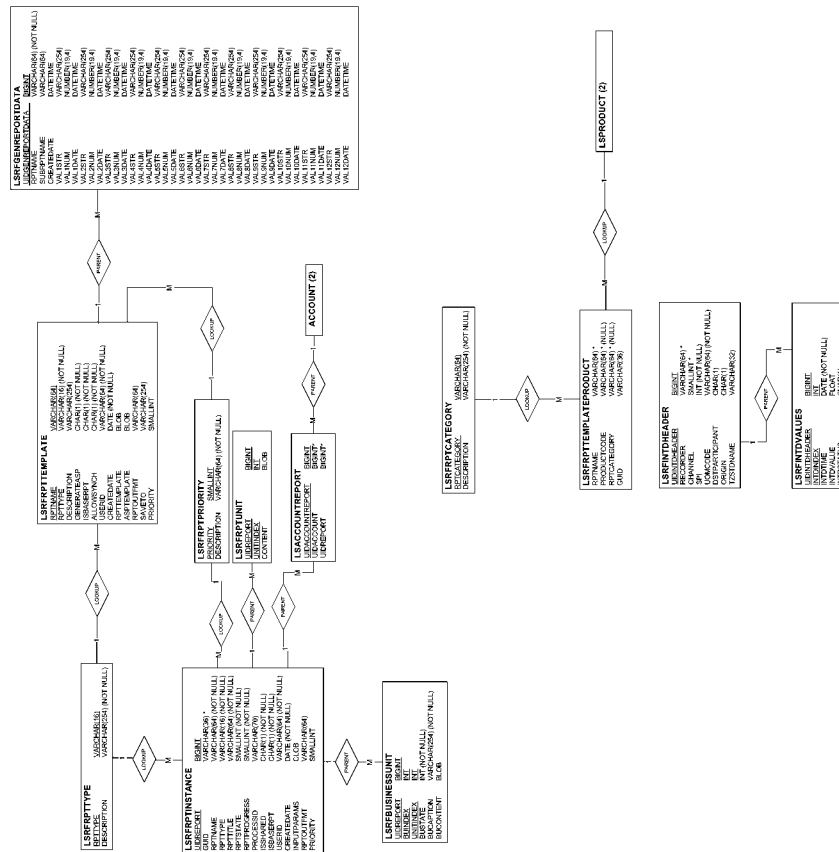






## Energy Information Platform Schema - Reporting

Oracle Utilities Energy Information Platform  
v1.6.0.0.0 – Reporting  
(Page 4 of 10)  
(Oracle Utilities Proprietary & Confidential)

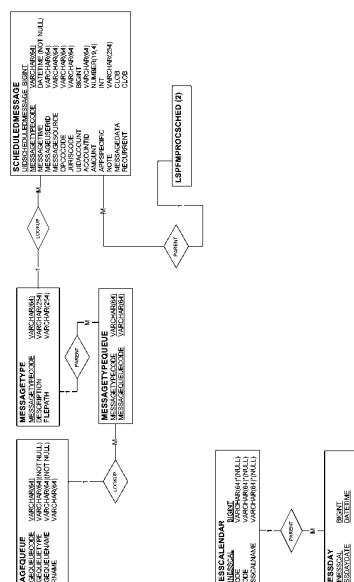
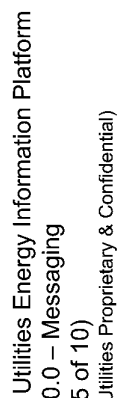


**Legend**

<b>Bold</b>	Table Name
<u>Underline</u>	Primary Key (Not Null ass.)
Single's(*)	Unique Index (Not Null)
String line - Entity	
Dashed line - Weak Entity	
Dashed line - Relationship	
Table Name (2)	Detailed Elsewhere

(Lookup Foreign Keys to OPERATINGCON and JURISDICTION are not shown)  
(If LAUSDIT schema extension enabled if on this page will have columns LUSER ar

## Energy Information Platform Schema - Messaging



### **Legend**

**Legend**  
**Bold - Table Name**

**bold** - Table Name  
Underline - Primary Key (Not Null assumed)

Asterisk(s)\*(") - Unique Index (Not Null assumed)

### Single line - Entity

Double line - Weak E

Dashed line - Relationship

Table Name (2) - Detailed Elsewhere

**Figure 1**

(Lookup Foreign Keys to OPERATING and ILLIBIDITION are not shown)

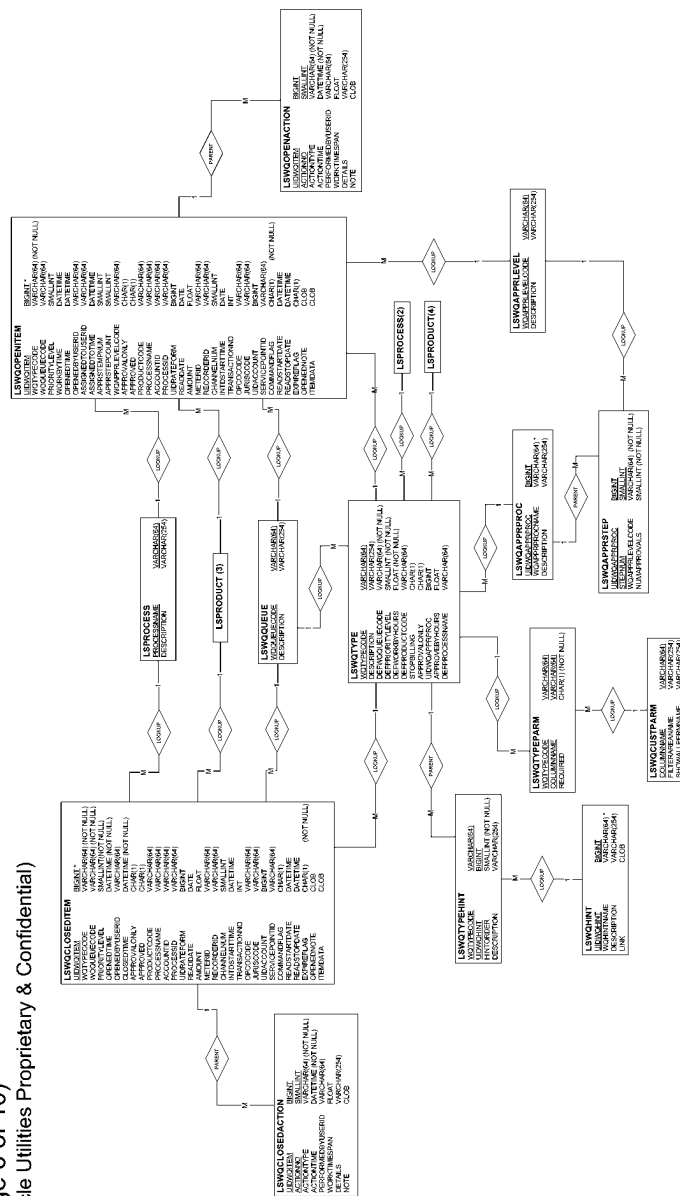
and JURISDICTION are not shown)

(If "LSAUDIT" schema extension enabled than all tables on this page will have columns LSUSER and LSTIME)

on this page will have columns LSUSER and LSTIME)

## Energy Information Platform Schema - Work Queues

Oracle Utilities Energy Information Platform  
v1.6.0.0 – Work Queues  
Page 6 of 10)  
Oracle Utilities Proprietary & Confidential)

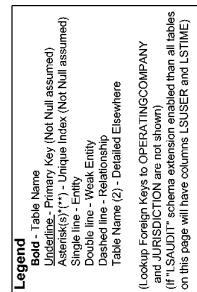


### Legend

**Legend**

- Bold** - Table Name
- Underline - Primary Key (Not Null assure)
- Asterisk(s) (\*) - Unique Index (Not Null)
- Single line - Entity
- Double line - Weak Entity
- Dashed line - Relationship
- Table Name (2) - Detailed Elsewhere

(Lookup Foreign Keys to OPERATINGCOMMI and JURISDICTION extension not shown)  
(If "SAUDIT" schema extension enabled the on this page will have columns LSUSER and





## Energy Information Platform Schema - Adapter

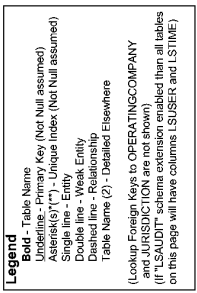


**Legend**

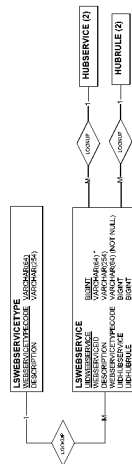
<b>Bold</b> - Table Name
<u>Underline</u> - Primary Key (Not Null ass.)
Angle's(*) - Unique Index (Not Null)
Single line - Entity
Double line - Weak Entity
Dashed line - Relationship
Table Name (2) - Detailed Elsewhere

(Lookup Foreign Keys to OPERATINGCO. and JURISDICTION are not shown) (#1, SAUDIT\* schema extension enabled) on this page will have columns LAUSER ar

## A-10 Energy Information Platform Configuration Guide



## Energy Information Platform Schema - Web Services



**Legend**

<b>Bold</b>	Table Name
<u>Underline</u>	Primary Key (Not Null assure Asterisk(s)***)
Single line - Entity	Unique Index (Not Null)
Double line - Weak Entity	
Dashed line - Relationship	
Table Name (2) - Detailed Elsewhere	

(Lookup Foreign Keys to OPERATING and JURISDICTION are not shown)  
(If LAUSDIT\* schema extension enabled then on this page will have columns LAUSER\* an



# Appendix B

---

## Oracle Utilities Data Repository Schema Import File Format

This appendix describes the format of the input files required by the Oracle Utilities import utilities, based on the base Oracle Utilities Data Repository schema (LSDB 5.31.002), including:

- **Record Formats**
  - **A Note Regarding Dependent Records**
- **Oracle Utilities Import Format (\*.imp)**
- **Oracle Utilities XML Import Format (\*.xml)**

### **Important Note**

Any client-specific customizations to this base schema, whether implemented by the client or Oracle, may alter one or more of the definitions provided below.

## Record Formats

Import files can be in one of two formats:

- **Oracle Utilities Import Format (\*.imp)**
- **Oracle Utilities XML Import Format (\*.xml)**

### A Note Regarding Dependent Records

Dependent records can only be added to the database after all of their parent records have been added. If a dependent record and one or more of its parent records are loaded from the same file, the parent records must appear before the dependent record in the file.

## Oracle Utilities Import Format (\*.imp)

The Oracle Utilities Import Format is a comma-delimited format in which each column value for a record must appear on a single line, in correct order, separated by commas. The following rules provide specific details concerning the input file format:

- Each record must begin on a new line (though text wrapping may occur).
- Comments may be placed on a line by themselves by beginning the line with an asterisk (\*) character. Blank lines are also allowed.
- The first value of each record must be the identifying mnemonic for the record type. The table layouts (see **How to view or print database table layouts:** on page B-3) contain the mnemonic for each table.
- Each value may optionally be enclosed by double-quotation marks. If a value is enclosed in double quotation marks, then all characters between the quotation marks will be considered part of the value, including any leading or trailing blanks.
- If a value contains a comma (,) character, or any leading or trailing blank characters, then it **must** be enclosed in double-quotation marks.
- Null values are represented by no value. Trailing null values must be preceded by commas.

### Example:

\*The following lines contain the Customer, Account Status, Jurisdiction, Operating Company, Revenue Code, and Account information.

```
CUST,"800001","Paul Pham",,"HOME","OFFICE1","E","E"
ACCTSTAT,"A","Excellent"
JURIS,"SF","Smithfield"
OPCO,"GECO","Generic Electric Company","123 Mars Planet
Ave","Universe","GK","12345-4321","991-123-4567"
REVCODE,"33","Commercial (Light)"
ACCT,"800001","800001","01/01/1993 00:00:00",,"GECO","SF","Paul
Sauvageau",,"A","33","2","3","N",5,5
```

Each table in the Oracle Utilities Data Repository has its own record format, based on the specific columns in the table. The PLIMPORT command line program can be used to view or output a layout of each table that defines its record format.

The -t switch on the PLIMPORT program allows users to view the layout of a table (or tables), or to create a text-only output file that contains the table layout. See **Importing Customer Data** on page 8-4 of the *Oracle Utilities Energy Information Platform Configuration Guide* for more information about using the PLIMPORT program.

**Note:** Using the -t switch overrides the other parameters. PLIMPORT can either import data or display tables in the database, but not both in the same command line. In other words, if you use the -t switch in the command line, PLIMPORT will NOT import data.

### How to view or print database table layouts:

1. Open a Command Prompt (DOS) Window. Select **Start->Run**, and type 'cmd' in the field that appears.
2. Change to the directory in which your Oracle Utilities executables are stored, typically \LODESTAR\BIN.
3. To view the database layout, type the following command at the prompt:

```
PLIMPORT -d dsn -t TABLENAME
```

where:

- The **-d** switch is identical to the switch described in **Importing Customer Data** on page 8-4 of the *Oracle Utilities Energy Information Platform Configuration Guide*.
  - The **-t** switch specifies the table you wish to view. To view the entire database, enter **-t** without the tablename parameter.
4. To save the table layouts to a text file, type the following command at the prompt:

```
PLIMPORT -d dsn -t TABLENAME > outputfile.txt
```

where:

- **outputfile.txt** is name of the output file containing the table layouts.

PLIMPORT writes the text-only output file containing the layout of the specified table(s) to the current directory, which can be viewed and printed using Notepad or another text editor.

For example, to save the layout of the Account Table in the “lodestar” database to a file named acctpict.txt, type the following

```
PLIMPORT -d lodestar -t ACCOUNT > acctpict.txt
```

The output might look like the following:

Column Name	Data Type	Data Length	Req'd?	Unique	Identity	Foreign Key
MNEMONIC	“ACCT”	N/A	X			
CUSTOMERID	character	64 (max)	X			1 - CUSTOMER
ACCOUNTID	character	64 (max)	X	1	X	
STARTTIME	datetime	N/A	X			
STOPTIME	datetime	N/A				
OPCOCODE	character	64 (max)				2 - OPERATINGCOMPANY
JURISCODE	character	64 (max)				3 - JURISDICTION
NAME	character	64(max)				
SIC	character	64 (max)				4 - SIC
ACCOUNTSTATUSCODE	character	64 (max)				5 - ACCOUNTSTATUS
REVENUECODE	character	64 (max)				6 - REVENUECODE
BILLINGMODEFLAG	character (0,1,2,3)	1				
PRINTDETAIL	character (0,1,2,3)	1				
FULLDAYBILL	character (0,Y, N)	1				
PREWINDOW	small integer	N/A				
POSTWINDOW	small integer	N/A				
EDI	character	1				
REGIONCODE	character	64 (max)				7 - REGION



## Rules

There are specific rules regarding the Oracle Utilities Import format, including:

- A single record will have at least one unique column (or compound column combinations), representing the primary key, or identity, of the record.
- The use of a tilde (“~”) in an identity column causes the column to be ignored. The identity of the record must still be discernible from the included identity columns.
- A record may have zero or more foreign key columns (or compound column combinations). The existence of a foreign key indicates a dependent record, meaning that all of its parent records must exist in the database before it is added. If a dependent record and one or more of its parent records are imported in the same file, the parent records must appear before the dependent record in the file. For example, before records can be added to the Account table in the above example, appropriate records must exist in the database in the Customer, Operating Company, Jurisdiction, SIC, Account Status, Revenue Code, and Region tables. If these records were imported using the same import file, the Customer, Operating Company, Jurisdiction, SIC, Account Status, Revenue Code, and Region table records would have to appear before the Account table records in the import file.
- If a record is added and one of the columns in the record is Null (i.e. there is no value for the column provided in the input file), the column is ignored. This allows default values in the database to remain unaffected.
- If updating a record, the new record must be followed **on the same line** by a comma and the primary key of the existing record (comma separated if more than one column).

### Example:

Update the jurisdiction record from Smithfield (SF) to California (CA).

```
JURIS,"CA","California",, "SF"
```

## Date Formats

*Billmonth* data types are formatted as “MM/YYYY”.

*Date* data types are formatted as either:

- “MM/DD/YYYY”
- “YYYY/MM/DD”

*Datetime (Timestamp)* data types are formatted as either:

- “MM/DD/YYYY HH:MM:SS”
- “YYYY/MM/DD HH:MM:SS”

**Note:** If import data contains milliseconds, the milliseconds must be set to “000” before importing.

## Oracle Utilities XML Import Format (\*.xml)

The Oracle Utilities XML Import Format is a XML format in which each record and column value for a record are defined in a specific XML structure. This is the same XML format used by the Oracle Utilities Database interface (LSImportDB). A Data Type Definition (DTD), xml examples, and data element descriptions for this format are provided below.

### DTD - Oracle Utilities XML Import Format

```
<!DOCTYPE LODESTAR_IMPORT
[
<!ELEMENT LODESTAR_IMPORT (ROWS*)>
<!ATTLIST LODESTAR_IMPORT
 ACTION#PCDATA,
 BATCHSIZE#PCDATA,
 ONERROR#PCDATA>
<!ELEMENT ROWS(ROW*)>
<!ATTLIST ROWS
 TBL #PCDATA>
<!ELEMENT ROW (DBCOLUMNNAME)>
<!ATTLIST ROW
 TBL CDATA #IMPLIED,
 ACTION(ADD|UPDATE|DELETE|ADDUPDATE)#IMPLIED>
<!ELEMENT DBCOLUMNNAME(#PCDATA)>
<!ATTLIST DBCOLUMNNAME
 V CDATA "column value",
 NV CDATA "new column value">
]>
```

### XML Examples - Oracle Utilities XML Import Format

#### Add Example

```
<LODESTAR_IMPORT ACTION="ADD" BATCHSIZE="50" ONERROR="CONTINUE">
 <ROWS>
 <ROW TBL="ACCOUNT">
 <CUSTOMERID V="Cust1"/>
 <ACCTNID V="Acct1"/>
 <STARTTIME V="01/01/2000 00:00:00"/>
 </ROW>
 </ROWS>
</LODESTAR_IMPORT>
```

#### Update Example

```
<LODESTAR_IMPORT ACTION="UPDATE" BATCHSIZE="50" ONERROR="STOP">
 <ROWS TBL="ACCOUNT">
 <ROW>
 <CUSTOMERID V="Cust1"/>
 <ACCTNID V="Acct1" NV="Acct100"/>
 <STARTTIME V="01/01/2001 00:00:00"/>
 </ROW>
 </ROWS>
</LODESTAR_IMPORT>
```

## Element Descriptions - Oracle Utilities XML Import Format

**LODESTAR\_IMPORT:** Root element containing one or more sets of records, each defined in a ROWS element.

Attributes:

**ACTION:** Defines the action performed. Possible values are:

- **ADD:** Adds one or more records to the database. It is an error if the record(s) is already present.
- **UPDATE:** Updates one or more existing records in the database. It is an error if the record does not exist.
- **DELETE:** Deletes one or more records in the database. It is an error if the record to be deleted does not exist.
- **ADDUPDATE:** Adds or Updates one or more records in the database. This is the Default behavior. If a record is present, the function updates it. If the record does not exist, the function adds the record.

**BATCHSIZE:** The number of records to be sent to the database in a single call. Larger numbers generally improve performance. The default is 50. The maximum is 500.

**ONERROR:** Defines how errors are handled. Possible values include:

- **CONTINUE:** Continue importing records after error.
- **STOP:** Stop importing records when an error occurs.

Elements:

**ROWS:** Element containing one or more records for import, each defined in a ROW element.

Attributes:

**TBL:** Table name for all the records defined in the ROW elements included in the ROW element.

Elements:

**ROW:** Element containing one record for import.

Attributes:

**TBL:** Table name for the record. If present in a ROW element, this overrides the TBL attribute on the ROWS element.

Elements:

**DBCOLUMNNAME:** Actual database name of the column. For example, when importing records in the Account ID column of the Account table, this would be "ACCOUNTID".

Attributes:

**V:** Column value. If not present, NULL is assumed. If the value is an XML, it will be specified as a child of the DBCOLUMNNAME element.

**NV:** New column value for the record. Must be specified if updating a key column.

**Notes:**

- The "NV" (New value) attribute is only required for identity columns that need to be updated.
- Also, for "DELETE" operations, providing just the identity columns is sufficient.
- Any missing "V" (Value) attribute will be treated as a NULL value for the column.



# Appendix C

---

## Oracle Utilities Enhanced Input/Output Interval Data Format

This document provides a detailed description of the Oracle Utilities Enhanced Input/Output Interval Data (LSE) format, used by Oracle Utilities applications. This includes:

- **Enhanced Format**
- **Enhanced Format Details**
- **Sample Files**
- **Units of Measure**

# Enhanced Format

## General Field Descriptions

In the enhanced format, fields are comma-delimited. A data line may contain as many full data-status-time groups as desired. No physical record in the file may exceed a character count (record length) of 32750. Leading or trailing white space is allowed on any field, but will be ignored upon processing. *The descriptor field is an exception to this rule.* It may contain any number of spaces, anywhere within the field.

Numeric fields will be expressed as decimal numbers, and may contain a decimal point and/or a leading minus sign (-), but must not contain commas or the value “+” to indicate a positive number. Non-negative numbers are assumed on all numeric fields if no leading minus sign is present.

The Customer Identifier, Channel Number, and the Start Time compose the full key of the record which, in combination with each other, must uniquely identify the record to be stored in the database.

No field except the descriptor may contain a comma.

All character values in the Enhanced Format must be uppercase.

Any cut entered with an invalid Start or Stop Time will be rejected.

An input cut with header records, but no data records, will be treated like any other cut; the number of intervals allocated will be determined by the time range, and all intervals will be treated as missing.

The contents of the Customer Identifier field, Origin field, and all Y/N indicators will be translated to uppercase before they are stored in the database.

**Note:** Database sort order will follow ASCII collating sequence, starting with the Customer Identifier and followed in turn by the Channel, then the Start Time.

A cut will be rejected if the number of interval values supplied is greater than the number of interval values expected, based upon the Start Time, Stop Time, SPI, and DST\_Participant fields. If too few interval values are supplied, the cut will be filled out with “missings” (value of 0 and status of ‘9’).

Any Stop Time falling exactly on an interval boundary will have one second subtracted from it.

## Field Relationships and Requirements

Relationships between fields exist in the enhanced format. These are described along with detailed field requirements in this section. Defaults are inserted when the processing program encounters an omitted field (indicated by double commas) or white space between commas.

### Sort Code

#### Relationships:

This field has no relationship with any other field.

#### Requirements:

- Sort codes must contain eight digits, including leading 0s. Sort codes are restricted to integers between 00000001 and 99999999. However, 00000005 through 09999999 are not used, and may be ignored (see below).
- Leading white space before the sort codes is not allowed.
- The first three header records are required.
- The fourth header record is optional.
- Header records will have sort code 00000001 through 00000004.
- Column 1 of each header record must contain a 0.
- Records with Sort codes 00000005 through 05000000 are reserved.
- Sort codes 00000030 through 00000039 are reserved for user-defined attributes created through Oracle Utilities Rules Language rate schedules.
- Data records must start with sort code 10000000, and progress sequentially (i.e. 10000001, 10000002...) through 99999999, or as high as needed to accommodate the data, with the maximum of 99999999.
- Column 1 of the data records must be greater than 0.

### Recorder/Customer Identifier (key element)

#### Relationships:

None.

#### Requirements:

- Only letters, numbers, underscores, and hyphens will be accepted in this field; all other characters and embedded blanks will cause the record to be rejected.
- Note:** Lowercase letters will be translated to uppercase before they are stored in the database.
- Must be at least one character, and no more than 64 characters in length.

#### Default:

None; field is required.

**Channel (key element)****Relationships:**

None.

**Requirements:**

- Field values must be non-negative integer only.
- Maximum value is 32767.

**Default:**

None; field is required.

**Start Time (key element)****Relationships:**

- Must be a valid datetime value less than the value of the Stop Time.

**Requirements:**

- Format must be YYYYMMDDHHMMSS.
- Value must be after 19700101000000 when using Data Manager, Oracle Utilities Billing Component, and other non-Oracle Utilities Load Analysis applications. The value must be after 19670101000000 when using Oracle Utilities Load Analysis.
- No spaces, colons, or any ASCII value allowed between time components.

**Default:**

None; field is required.

**Stop Time****Relationships:**

- Must be a valid datetime value greater than the value of the Start Time.
- Value must agree with the number of interval values past the start time \* SPI.
- One second is subtracted if the value is equal to an even interval start time.

**Requirements:**

See Start Time.

**Default:**

None; field is required

**DST Participant Flag****Relationships:**

None.

**Requirements:**

Valid values are Y, N, or A, indicating whether or not this record will be processed using DST adjustments.

---

Flag Value	Definition
Y	(DST Participant) The intervals recorded participate in DST adjustments and the intervals during DST reflect the time change.

---



Flag Value	Definition
N	(Does not participate in DST) The intervals recorded do not make any DST time changes.
A	(DST participant) The intervals have been adjusted to 24-hour days. (At the April time change, there should be a value at the time change (a place holder) of 0 with status code “9”. At the October time change, there should be a combined value.) These cuts will be converted to a “Y” or “N”, and are not stored in the database as “A”.

**Default:**

As specified in the INTDDEFAULTDST configuration parameter setting, the INTDCONFIG.XML file, or the CSDST value in Oracle Utilities Load Analysis.

**Invalid Record Flag****Relationships:**

None.

**Requirements:**

Valid values are Y or N, indicating whether or not the input programs should validate the incoming record.

Flag Value	Definitions
Y	The incoming record contains unvalidated data. In Oracle Utilities Load Analysis, a value of Y can be used to run this record through the validation routines.
N	The incoming record contains valid data (mark as valid).

**Default:**

Y.

**Meter Start Reading****Relationships:**

Must be supplied if any of the other Meter fields (Meter Stop Reading, Meter Multiplier, or Meter Offset) are supplied.

**Requirements:**

If supplied, value must be a non-negative numeric value.

**Default:**

0.

**Meter Stop Reading****Relationships:**

Must be supplied if any of the other Meter Fields (Meter Start Reading, Meter Multiplier, or Meter Offset) are supplied.

**Requirements:**

If supplied, value must be a non-negative numeric value.

**Default:**

0 stored in the database if not supplied.

**Meter Multiplier**

**Relationships:**

Must be supplied if any of the other Meter fields (Meter Start Reading, Meter Stop Reading, or Meter Offset) are supplied.

**Requirements:**

If supplied, value must be positive numeric. However, if no value is specified, 0 will be stored. Zero (0) is not a valid value for the meter multiplier, but its presence in the database indicates that no meter multiplier was supplied.

**Default:**

None, 0 stored in the database if not supplied.

**Note:** If 0 is stored and/or observed in the database and a cut is exported to the LSE format, all meter fields for that cut will be absent. A zero stored in the database is indicative of “no meter information supplied”.

**Meter Offset**

**Relationships:**

Must be supplied if any of the other Meter fields (Meter Start Reading, Meter Stop Reading, or Meter Multiplier) are supplied.

**Requirements:**

If supplied, value must be numeric.

**Default:**

0 stored in the database if not supplied

**Pulse Multiplier**

**Relationships:**

None.

**Requirements:**

Value must be positive numeric. Also, must be supplied if Pulse Offset (see below) is supplied.

**Default:**

None, 0 stored in the database if not supplied.

**Note:** If 0 is stored and/or observed in the database and a cut is exported to the LSE format, all pulse fields for that cut will be absent. A zero stored in the database is indicative of “no pulse information supplied”.

## Pulse Offset

### Relationships:

If supplied, you must also supply Pulse Multiplier.

### Requirements:

Value must be numeric.

### Default:

None. (Zero is stored in the database to indicate that the Pulse Multiplier was not supplied.)

## Seconds Per Interval (SPI)

### Description:

This field stores the interval duration if appropriate. An Intervals Per Hour (IPH) of 1 can be translated to a Seconds Per Interval of 3600.

### Relationships:

Value of 0 indicates non-uniform recording duration and a full array of interval start values (one per recording) located in the data section. (This feature will be supported in a future release and is **not currently available**.)

### Requirements:

- Value must be a positive integer.
- Valid values for SPI are 86400, 3600, 1800, 900, 300, and 60.

### Default:

None; field is required.

## Unit of Measure

### Description:

See **Units of Measure** on page C-16 for valid Units of Measure.

### Relationships:

None.

### Requirements:

- Value should be a valid unit of measure. If the value entered is unknown to, 16 (Miscellaneous) will be stored.
- Value must be non-negative numeric.
- Maximum value is 32767.

### Default:

None; field is required.

## Basic Unit Code

### Description:

This field is reserved for future use.

### Relationships:

None.

### Requirements:

Omit this field.

### Default:

0.

## Times Zones West of GMT

### Description:

This value tells programs accessing this record which time zone the values have been recorded in. Its value will be the difference in time, heading west, between Greenwich Mean Time and the represented time zone as measured in half hours. Eastern Standard Time should have a value of 10; Pacific = 16. A Value of -1 will indicate "Time Zone not available".

### Relationships:

None.

### Requirements:

Value must be -1 or a non-negative numeric between 0 and 47, that represents the time zones West of GMT measured in half-hour increments. Any record containing a time zone greater than 47 will be rejected.

### Default:

-1.

## Population

### Description:

For Oracle Utilities Load Analysis statistical records only.

### Relationships:

None.

### Requirements:

Value may be any non-negative integer value.

### Default:

0.

---

## Weight

**Description:**

For Oracle Utilities Load Analysis statistical records only.

**Relationships:**

None.

**Requirements:**

Value may be any non-negative numeric value.

**Default:**

0.0.

## Time Zone Standard Name

**Description:**

Defines the Time Zone to which the cut is associated. The value must be one of EST, CST, MST, PST, or be defined in the LSCALENDAR.XML configuration file (if present). Each value in this field must contain printable ASCII characters no longer than 32 bytes.

Only letters, numbers, underscores, and hyphens will be accepted in this field, all other characters and embedded blanks will cause the record to be rejected.

**Relationships:**

Must map exactly to one of EST, CST, MST, PST or an entry in the LSCALENDAR.XML file (if present). Oracle Utilities Load Analysis will import without checking this file.

**Requirements**

Length: <= 32 bytes

## Descriptor

**Description:**

The user may enter any descriptive information in this field. Any information entered into this field will be stored in the database verbatim as supplied.

**Relationships:**

None.

**Requirements:**

- Value must be 80 characters or fewer, and may contain commas.
- May start with blanks.
- If more than 80 characters are entered, truncation will occur and the first 80 will be used.

## Timestamp

### Relationships:

None.

### Requirements:

- Format should be YYYYMMDDHHMMSSmmm. The milliseconds (mmm) may be omitted, in which case mmm will be set to 000.
- No spaces, colons, or any ASCII value allowed between time components.

### Default:

Current time that data is loaded into database.

**Note:** It is strongly recommended that you leave this field empty and allow the system to input the TimeStamp for you.

## Origin

### Relationships:

None.

### Requirements:

Must be one of: M (metered), P (profiled), C (computed), or S (Oracle Utilities Load Analysis Statistic).

### Default:

M (metered).

## Interval Value

### Description:

This field contains the actual recorded value for a time period.

### Relationships:

Must have one status code entry per interval data value.

### Requirements:

Any numeric value. (Up to 15 significant digits can be stored.)

### Default:

0 (see Status Code, Relationships, below).

## Status Code

### Description:

This field contains the status code for the preceding Interval Value.

### Relationships:

Must have an entry for the Interval Value.

### Requirements:

Any character will be accepted.

### Default:

- This field defaults to a blank (‘ ’) when omitted if the interval value is present.
- This field defaults to ‘9’ when the Interval Value field is omitted.

## Interval Start

### Description:

- This field indicates the starting time of the recording or the time when the recording took place.
- Format must be YYYYMMDDHHMMSS. (This feature will be supported in a future release and is **not currently available**.)

### Relationships:

- Until this feature is supported, the SPI may not be omitted (represented by „).
- There must be one entry per Interval Value.

### Requirements:

See Start Time, Requirements.

### Default:

None.

## Enhanced Format Details

### First Header Record Format

ELEMENT	DESCRIPTION	MAXIMUM LENGTH IN BYTES (if applicable)	COMMENT
1	Sort Code	8	Must be 00000001
2	Recorder/ Customer Identifier	64	Letters, numbers, hyphens or underscores are acceptable Values
3	Channel	5	Max is 32767
4	Start Time	14	YYYYMMDDHHMMSS (24-hour)
5	Stop Time	14	YYYYMMDDHHMMSS (24-hour)
6	DST Participant Flag	1	Y/N/A
7	Invalid Record Flag	1	Y/N

### Second Header Record Format

ELEMENT	DESCRIPTION	MAXIMUM LENGTH IN BYTES (if applicable)	COMMENT
1	Sort Code	8	Must be 00000002 Default: None; field required
2	Meter Start Reading	N/A	Non-Negative Numeric Max: 999999999999999.9999 Default: 0
3	Meter Stop Reading	N/A	Non-Negative Numeric Max: 999999999999999.9999 Default: 0
4	Meter Multiplier	N/A	Positive Numeric (Optional) Max: 999999999999999.9999 Default: 0
5	Meter Offset	N/A	Numeric (Optional) Max: 999999999999999.9999 Default: 0
6	Pulse Multiplier	N/A	Positive Numeric (Optional) Max: 999999999999999.9999 Default: 0
7	Pulse Offset	N/A	Numeric (Optional) Max: 999999999999999.9999 Default: None; field required
8	Seconds Per Interval (SPI)	N/A	Positive Numeric Default: None; field required



ELEMENT	DESCRIPTION	MAXIMUM LENGTH IN BYTES (if applicable)	COMMENT
9	Unit of Measure	N/A	Numeric Max: 32767 Default: None; field required
10	Basic Unit Code	N/A	Positive Numeric (Optional) Max: 9999 Default: 0
11	Time Zones West of GMT	N/A	Numeric. (Optional) Default: -1 Min: -1 Max: 47
12	Population	N/A	Positive Numeric (Optional) Max: 999999999999999.9999 Default: 0.0
13	Weight	N/A	Positive Numeric (Optional) Max: 999999999999999.9999 Default: 0.0
14	Time Zone Standard Name		Character (32) Overrides any value in TZWGMT. Must be one of "EST", "CST", "MST", "PST", or definition must exist in LSCalendar.xml file.

### Third Header Record Format

ELEMENT	DESCRIPTION	MAXIMUM LENGTH IN BYTES (if applicable)	COMMENT
1	Sort Code	8	Must be 00000003
2	Descriptor	80	

### Fourth Header Record Format

ELEMENT	DESCRIPTION	MAXIMUM LENGTH IN BYTES (if applicable)	COMMENT
1	Sort Code	8	Must be 00000004
2	Timestamp	17	YYYYMMDDHHMMSSMM (optional) Default: current date
3	Origin	1	C, M, P, or S (optional)

Data Record Format

ELEMENT	DESCRIPTION	MAXIMUM LENGTH IN BYTES (if applicable)	COMMENT
1	Sort Code	8	10000000 through 99999999
2	Interval Value		
3	Status Code	1	
4	Interval Start	14	YYYYMMDDHHMMSS (optional)

A data line may contain as many full data-status-time groups as desired, providing the record does not exceed a character count (record length) of 32750.

Interval start times may be omitted, but a comma must still be present as a placeholder. Data must be contiguous and “missings” (or gaps) must be represented by value 0 and status '9'. At this time, all intervals must be of the same duration, hence Interval Start Times should not be specified.

## Sample Files

### Enhanced Direct Input File

A sample input file using the enhanced format with 3 headers and 2 data records and with interval starts omitted:

```
00000001,N1732,17,19820412000001,19820412005959,Y,Y
00000002,23887.34,23903.56,1.0,0.0,0.0144,0.0,900,1,0,-1,,,
00000003,ACCEPTANCE TEST DATA NUMBER 1
00000004,,M
10000000,31,Q,,9.52,R,,.00314,S,,123456.98765,T,,
10000001,32,Q,,9.53,R,,.00315,S,,123457,T,,
```

### Standard Direct Input File

A sample old-format file is provided below for comparison.

```
0001WARNINGTEST 10412821351051382140004010
0002013127501400250000000000912350000000000945679-000000000012346-
0000000000034567
0003ACCEPTANCE TEST DATA 00000000000000000000000000000000
0004
100000031Q00028Q00008Q00061Q00038Q00011Q00050Q00005Q00055Q00000900000900004Q
100100029Q00016Q00079Q00036Q00020Q00076600061600015600078Q00021Q00050Q00004Q
100200004Q00007Q00006Q00006Q00004Q00008Q00005Q00009Q00009Q00009Q00005Q00006Q
```

## Units of Measure

The table below lists valid Units of Measure. Intervals may be summed or averaged when aggregating, depending on the unit of measure. The aggregation technique is noted on the table

**Note:** Oracle Utilities products also accept 3-digit UOMs, which may be user-defined.

CODE	DESCRIPTION	AGGREGATION TECHNIQUE
01	kWh	Sum
02	kW	Average
03	kVARh	Sum
04	kVAH	Sum
05	Temperature (°F)	Average
06	KQD	Average
07	V <sup>2</sup> H (PTP)	Sum
08	kQh	Sum
09	kQh (45 degrees)	Sum
10	I <sup>2</sup> H	Sum
11	Volts	Average
12	Amps	Average
13	Temperature (°C)	Average
14	Dewpoint	Average
15	Amplitude	Average
16	Miscellaneous	Sum
17	Minute Run Time (MRT)	Sum
18	Wind Velocity (Cm/Sec)	Average
19	Fraction V2H (PTN)	Average
20	Percent	Average
21	Flow	Average
22	kVAR	Average
23	kVA	Average
24	kVA Ratio	Average
25	Power Factor	Average
26	Hertz	Average
27	Feet	Average

CODE	DESCRIPTION	AGGREGATION TECHNIQUE
28	Minutes	Sum
29	On / Off (Tap position)	Average
30	Inches	Average
31	Individual kWh	Sum
32	kWh r	Sum
33	Individual Totalized kVARh	Sum
34	kVARh r	Sum
35	Individual Totalized Temperature (°F)	Average
36	kVAh r	Sum
37	IND V <sup>2</sup> H (Individual totalized V <sup>2</sup> H)	Sum
38	IND kQh (Individual totalized kQh)	Sum
39	KQH r	Sum
40	Miscellaneous	Average
41	IND Volts (Individual totalized Volts)	Average
42	IND Amps (Individual totalized Amperes)	Average
43	IND Temperature (°C) (Individual totalized temperature, degrees Celsius)	Average
44	Mw	Sum
45	MVAR	Average
46	MVA	Average
47	IND MRT (Individual totalized MRT)	Sum
48	IND CMS (Individual totalized CMS)	Average
49	Run Hours	Sum
50	Equivalent Full Load Hours	Sum
51	KWH-OUT	Sum
52	KW-OUT	Average
53	KVARH-OUT	Sum
54	KVAH-OUT	Sum
55	KQH-OUT	Sum
56	LEAD-KVARH	Sum
57	LEAD-KVARH-OUT	Sum
58	LAG-KVARH	Sum

CODE	DESCRIPTION	AGGREGATION TECHNIQUE
59	LAG-KVARH-OUT	Sum
60	Gallons / Minute	Average
61	BTU	Sum
62	THERMS	Sum
63	Cubic Feet / Minute	Average
64	Cubic Feet / Second	Average
65	WM <sup>2</sup>	Average
66	Relative Humidity	Average
67	MPH	Average
68	THI	Average
69	Gallons	Sum
70	Cubic Feet	Sum
71	Temperature Difference	Average
72	KVAR-OUT	Average
73	KVA-OUT	Average
74	Knots	Average
75	Degrees	Average
76	CCF (Hundred cubic feet)	Sum
77	CF / Hour	Average
78	Pounds / Square inch	Average
79	Dollars	Sum
80	DECATHERMS	Sum
81	Pounds	Sum
82	Pounds / Hour	Average
83	MPOUNDS	Sum
84	MPOUNDS / Hour	Average
85	\$/KWH	Average
86	\$/MW	Average
87	\$/MWH	Average
88	\$/Hour	Average
89	Volt Hours	Sum
90	Individual Totalized Cubic Feet	Sum

CODE	DESCRIPTION	AGGREGATION TECHNIQUE
91	Individual Totalized Btu	Sum
92	Pressure in MILLIBARS	Average
93	Visibility in Miles	Average
94	Cents per kWh	Average
99	Individual Totalized Gallons	Sum
100	MWH	Sum
102	Euros	Sum
103	Euros per MWH	Average
104	Euros MW	Average
105	GW	Average
106	TWH	Sum
107	Cubic Meters (M3)	Sum
108	Mega Joules per Cubic Meter (MJ/m3)	Average
109	Kilograms per Cubic Meter (Kg/m3)	Sum
110	Cubic Meters per Hour (M3/H)	Average

### UOM Compatibility

When merging cuts with different units-of-measure, cuts with specific differing UOMs can be combined while others can't. For example, you can't merge two cuts if the UOM of one cut is inches and the UOM of the other is dollars, because whichever UOM is assigned to the merged cut wouldn't apply to at least some of the data.

However, cuts with certain specific different UOMs can be combined. UOMs that can be combined are referred to as compatible UOMs, and are listed in the table below.

Compatible UOMs
01,31,32,51
02,52
03,33,34,53,56,57,58,59
04,36,54
05,35
07,37
08,09,38,39,55
11,41
12,42
13,43

Compatible UOMs
17,47
18,48
22,72
23,73
61,69
69,99
70,90

These units of measures may have their intervals divided by the IPH for display:

Demand-Type UOMs
02, 05, 06, 07, 10, 22, 23, 24, 52, 72, 73, 105, 110



# Appendix D

---

## Oracle Utilities Standard XML Interval Data Format

This appendix provides a detailed description of the Oracle Utilities Standard XML Interval Data Formats, supported by Oracle Utilities applications. This includes:

- **Standard XML Format**
- **XML Standard File Format Example**
- **XML Standard File Format DTD**
- **XML Compact Format**

# Standard XML Format

## General Node Descriptions

In the Oracle Utilities Standard XML format, nodes are enclosed with XML tags. A data line may contain as many full data-status-time groups as desired.

Numeric nodes will be expressed as decimal numbers, and may contain a decimal point and/or a leading minus sign (-). Non-negative numbers are assumed on all numeric nodes if no leading minus sign is present. If a minus sign is present, spaces may exist between it and the numeric value. For example: ..., - 122.11,... or ..., -23.3,... are valid entries for a numeric node.

The Recorder, Channel Number, and the Start Time compose the full key of the record which, in combination with each other, must uniquely identify the record to be stored in the database.

Node names and XML tag names must be uppercase.

Any cut entered with an invalid Start or Stop Time will be rejected.

The contents of the Recorder node, Origin node, and all Y/N indicators will be translated to uppercase before they are stored in the database.

**Note:** Database sort order will follow ASCII collating sequence, starting with the Recorder and followed in turn by the Channel, then the Start Time.

A cut will be rejected if the number of interval values supplied is greater than the number of interval values expected, based upon the Start Time, Stop Time, SPI, and DST\_Participant nodes. If too few interval values are supplied, the cut will be filled out with “missing” (value of 0 and status of ‘9’).

Any Stop Time falling exactly on an interval boundary will have one second subtracted from it.

## General Rules

Each root node of a file starts with a set of special nodes. These nodes contain the name of the format, the version number of the format, the decimal point specification, and date/time format specification.

### Format Name

The first node identifies the format as “LODESTAR Interval Data XML Format”. It is an error if this is not present.

#### Example:

```
<INTERVAL_DATA_FORMAT>LODESTAR Interval Data XML Format </INTERVAL_DATA_FORMAT>
```

### Format Version Number

The second node identifies the version number of the format. It is an error if this is not present.

#### Example:

```
<VERSION>version_number</VERSION>
```

### Decimal Point Specification

The third node identifies the decimal point specification to be used in the file. All float values used throughout the file must use this decimal point format. It is an error if this is not present.

#### Example:

```
<DECIMAL_POINT>.</DECIMAL_POINT>
```

### Date/Time Format Specification

The fourth node identifies the date/time format to be used in the file. There are three available date/time formats:

1. MM/DD/YYYY HH:MM:SS
2. DD/MM/YYYY HH:MM:SS
3. YYYY/MM/DD HH:MM:SS

All date/time values used throughout the file must use this format. It is an error if this is not present.

#### Example:

```
<DATETIME_FORMAT>10/27/1999 10:31:00</DATETIME_FORMAT>
```

These nodes are followed by the interval cut information as outlined below:

## Node Relationships and Requirements

Relationships between nodes exist in the standard XML format. These are described in this section, and detailed node requirements are provided. Defaults are inserted when the processing program encounters an omitted node.

### RECORDER (key element)

**Relationships:**

None.

**Requirements:**

- Only letters, numbers, underscores, and hyphens will be accepted in this node; all other characters and embedded blanks will cause the record to be rejected.

**Note:** Lowercase letters will be translated to uppercase before they are stored in the database.

- Must be at least one character, and no more than 64 characters in length.

**Default:**

None; node is required.

### CHANNEL (key element)

**Relationships:**

None.

**Requirements:**

- Node values must be non-negative integer only.
- Maximum value is 32767.

**Default:**

None; node is required.

### STARTTIME (key element)

**Relationships:**

- Must be a valid datetime value less than the value of the Stop Time.

**Requirements:**

- Format must match that specified by the **Date/Time Format Specification** on page D-3.
- Value must be after 19700101000000.
- No spaces, colons, or any ASCII value allowed between time components.

**Default:**

None; node is required.

## STOPTIME

### Relationships:

- Must be a valid datetime value greater than the value of the Start Time.
- Value must agree with the number of interval values past the start time \* SPI.
- One second is subtracted if the value is equal to an even interval start time.

### Requirements:

See STARTTIME.

### Default:

None; node is required.

## DST\_PARTICIPANT

### Relationships:

None.

### Requirements:

Valid values are Y, N, or A, indicating whether or not this record will be processed using DST adjustments.

Flag Value	Definition
Y	(DST Participant) The intervals recorded participate in DST adjustments and the intervals during DST reflect the time change.
N	(Does not participate in DST) The intervals recorded do not make any DST time changes.
A	(DST participant) The intervals have been adjusted to 24-hour days. (At the April time change, there should be a value at the time change (a place holder) of 0 with status code "9". At the October time change, there should be a combined value.) These cuts will be converted to a "Y" or "N", and are not stored in the database as "A".

### Default:

As specified in the INTDDEFAULTDST configuration parameter setting, the INTDCONFIG.XML file, or the CSDST value in Oracle Utilities Load Analysis.

## VALIDATION\_REQUIRED

**Relationships:**

None.

**Requirements:**

Valid values are Y or N indicating whether or not the record should be marked as valid.

Flag Value	Definition
Y	The incoming record contains unvalidated data.
N	The incoming record contains valid data, and should be marked as such.

**Default:**

Y.

## METER\_START\_READING

**Relationships:**

Must be supplied if any of the other Meter nodes (Meter Stop Reading, Meter Multiplier, or Meter Offset) are supplied.

**Requirements:**

If supplied, value must be a non-negative numeric value.

**Default:**

0.

## METER\_STOP\_READING

**Relationships:**

Must be supplied if any of the other Meter nodes (Meter Start Reading, Meter Multiplier, or Meter Offset) are supplied.

**Requirements:**

If supplied, value must be a non-negative numeric value.

**Default:**

0 (not supplied).

## METER\_MULTIPLIER

**Relationships:**

Must be supplied if any of the other Meter nodes (Meter Start Reading, Meter Stop Reading, or Meter Offset) are supplied.

**Requirements:**

If supplied, value must be positive numeric. However, if no value is specified, 0 will be stored. Zero (0) is not a valid value for the meter multiplier, but its presence in the database indicates that no meter multiplier was supplied.

**Default:**

None, 0 stored in the database if not supplied.

**Note:** If 0 is stored and/or observed in the database and a cut is exported to the LSE format, all meter fields for that cut will be absent. A zero stored in the database is indicative of “no meter information supplied”.

**METER\_OFFSET****Relationships:**

Must be supplied if any of the other Meter nodes (Meter Start Reading, Meter Stop Reading, or Meter Multiplier) are supplied.

**Requirements:**

If supplied, value must be numeric.

**Default:**

0 (not supplied).

**PULSE\_MULTIPLIER****Relationships:**

None.

**Requirements:**

Value must be positive numeric. Also, must be supplied if Pulse Offset (see below) is supplied.

**Default:**

None, 0 stored in the database if not supplied.

**Note:** If 0 is stored and/or observed in the database and a cut is exported to the LSE format, all pulse fields for that cut will be absent. A zero stored in the database is indicative of “no pulse information supplied”.

**PULSE\_OFFSET****Relationships:**

If supplied, you must also supply Pulse Multiplier.

**Requirements:**

Value must be numeric.

**Default:**

None.

**SPI (Seconds per Interval)****Description:**

This node will store the interval duration if appropriate. An Intervals per Hour (IPH) of 1 can be translated to an SPI of 3600.

**Relationships:**

Value of 0 indicates non-uniform recording duration and a full array of interval start values (one per recording) located in the data section. (This feature will be supported in a future release and is **not currently available**.)

**Requirements:**

- Value must be a positive integer.
- Valid values for SPI are 86400, 3600, 1800, 900, 300, and 60.

**Default:**

None; node is required.

**UOM (Unit of Measure)****Description:**

See **Units of Measure** on page B-16 for valid Units of Measure.

**Relationships:**

None.

**Requirements:**

- Value should be a valid unit of measure. If the value entered is unknown to, 16 (Miscellaneous) will be stored.
- Value must be non-negative numeric.
- Maximum value is 32767.

**Default:**

None; node is required.

**DC\_FLOW****Description:**

Power flow direction. This is used with MV90 data.

**Relationships:**

None.

**Requirements:**

Must be either “D” (Delivered) or “R” (Received).



**Default:**

“D”

**TIMEZONE****Description:**

This value tells programs accessing this record which time zone the values were recorded in. Its value will be the difference in time, heading west, between Greenwich Mean Time and the represented time zone, measured in half hours. Eastern Standard Time should have a value of 10; Pacific = 16. A Value of -1 indicates “Time Zone not available.”

**Relationships:**

None.

**Requirements:**

Value must be -1 or a non-negative numeric between 0 and 47, that represents the time zones west of GMT, measured in half-hour increments. Any record containing a time zone greater than 47 will be rejected.

**Default:**

-1.

**POPULATION****Description:**

For Oracle Utilities Load Analysis Statistical records only.

**Relationships:**

None.

**Requirements:**

Value may be any non-negative integer value.

**Default:**

0.

**WEIGHT****Description:**

For Oracle Utilities Load Analysis Statistical records only.

**Relationships:**

None.

**Requirements:**

Value may be any non-negative numeric value.

**Default:**

0.0.

## TIME\_ZONE\_STANDARD\_NAME

### Description:

Defines the Time Zone to which the cut is associated. The value must be one of EST, CST, MST, PST, or be defined in the LSCALENDAR.XML configuration file. Each value in this field must contain printable ASCII characters no longer than 32 bytes.

Only letters, numbers, underscores, and hyphens will be accepted in this field, all other characters and embedded blanks will cause the record to be rejected.

**Note:** Lowercase letters will be translated to uppercase before being stored

### Relationships:

Must map exactly to one of EST, CST, MST, PST or an entry in the LSCALENDAR.XML file. Oracle Utilities Load Analysis will import without checking this file.

### Requirements

Length: <= 32 bytes

## DESCRIPTOR

### Description:

The user may enter any descriptive information in this node. Any information entered into this node will be stored in the database verbatim.

### Relationships:

None.

### Requirements:

- Value must be 80 characters or fewer, and may contain commas.
- May start with blanks.
- If more than 80 characters are entered, truncation occurs and the first 80 are used.

## TIMESTAMP

### Relationships:

None.

### Requirements:

- Format must match that specified by the **Date/Time Format Specification** on page D-3.
- No spaces, colons, or any ASCII value allowed between time components.

### Default:

Current time that data is loaded into database.

**Note:** It is strongly recommended that you leave this node empty and allow the system to input the TimeStamp for you.

## ORIGIN

### Relationships:

None.

### Requirements:

Must be one of: M (metered), P (profiled), C (computed), or S (Oracle Utilities Load Analysis Statistic).

**Default:**

M (metered).

**ACCEPT\_REJECT\_STATUS****Description:**

This is used by MV90 to indicate if the cut was accepted or rejected.

**Relationships:**

None.

**Requirements:**

Will be either:

Status Code	Description
00	Accept. No Error.
01 though 07	Rejected/Reject type.

**Default:**

None.

**TRANSLATION DATE****Description:**

This is used by MV90 to indicate the date/time the cut was translated from MV90.

**Relationships:**

None.

**Requirements:**

Format must match that specified by the **Date/Time Format Specification** on page D-3).

**Default:**

None.

## Interval Values

The interval values data is enclosed within the <INTERVAL> node. Each individual interval value is enclosed within the <RECORD> node, and consists of the VALUE, STATUS (optional), and STARTTIME (optional). The RECORD node will not have a STATUS node if and only if the interval status is a space. The optional STARTTIME node in the RECORDING node is only included if the XMLEXPORTINTDSTARTTIME=1 parameter is included in the LODESTAR.CFG file (see **Chapter Four: Configuration Files** in the *Oracle Utilities Energy Information Platform Installation and Configuration Guide*).

Descriptions of the RECORD nodes are as follows:

### VALUE (Interval Value)

**Description:**

This node contains the actual recorded value for a time period.

**Relationships:**

Must have one status code entry per interval data value, if the status is not a space (see above).

**Requirements:**

Any numeric value. (Up to 15 significant digits can be stored.)

**Default:**

0 (see Status Code, Relationships, below).

### STATUS (Status Code)

**Description:**

This node contains the status code for the preceding Interval Value.

**Relationships:**

Must have an entry for the Interval Value.

**Requirements:**

Any character will be accepted.

**Default:**

- This node defaults to a blank (‘ ’) when omitted if the interval value is present.
- This node defaults to ‘9’ when the Interval Value node is omitted.

## STARTTIME (Interval Start Time)

### Description:

- This node indicates the starting time of the recording or the time when the recording took place.
- Format must match that specified by the **Date/Time Format Specification** (page D-3).
- This feature will be supported in a future release and is **not currently available**.

### Relationships:

- There can be one entry per Interval Value.

### Requirements:

See Start Time, Requirements.

### Default:

None.

# XML Standard File Format Example

An example of the Oracle Utilities Standard XML format is shown below.

```
<INTERVAL_DATA>
 <INTERVAL_DATA_FORMAT>LODESTAR Interval Data XML Format</INTERVAL_DATA_FORMAT>
 <VERSION>1.2</VERSION>
 <DECIMAL_POINT>.</DECIMAL_POINT>
 <DATETIME_FORMAT>MM/dd/yyyy HH:mm:ss</DATETIME_FORMAT>
 <CUT>
 <RECORDER>TV011285</RECORDER>
 <CHANNEL>32767</CHANNEL>
 <STARTTIME>01/01/1999 00:00:00</STARTTIME>
 <STOPTIME>01/31/1999 23:59:59</STOPTIME>
 <DST_PARTICIPANT>Y</DST_PARTICIPANT>
 <VALIDATION_REQUIRED>N</VALIDATION_REQUIRED>
 <METER_START_READING>1234</METER_START_READING>
 <METER_STOP_READING>2345</METER_STOP_READING>
 <METER_MULTIPLIER>1.12345</METER_MULTIPLIER>
 <METER_OFFSET>100.0</METER_OFFSET>
 <PULSE_MULTIPLIER>1.0000</PULSE_MULTIPLIER>
 <PULSE_OFFSET>0.0000</PULSE_OFFSET>
 <SPI>900</SPI>
 <UOM>01</UOM>
 <DC_FLOW>D</DC_FLOW>
 <TIMEZONE>8</TIMEZONE>
 <POPULATION>300000</POPULATION>
 <WEIGHT>0.0</WEIGHT>
 <TIME_ZONE_STANDARD_NAME>EST</TIME_ZONE_STANDARD_NAME>
 <DESCRIPTOR>DEMO</DESCRIPTOR>
 <TIMESTAMP>01/01/2000 00:00:00</TIMESTAMP>
 <ORIGIN>M</ORIGIN>
 <INTERVAL>
 <RECORDING>
 <VALUE>123456.1234</VALUE>
 <STATUS>A</STATUS>
 <STARTTIME>11/01/1999 11:12:34</STARTTIME>
 </RECORDING>
 <RECORDING>
 <VALUE>123456.1234</VALUE>
 <STATUS>A</STATUS>
 <STARTTIME>11/01/1999 11:15:00</STARTTIME>
 </RECORDING>
 </INTERVAL>
 </CUT>
</INTERVAL_DATA>
```

# XML Standard File Format DTD

The Data Type Definition for the Oracle Utilities Standard XML format is shown below.

```
<!DOCTYPE INTERVAL_DATA
[
<!ELEMENT INTERVAL_DATA (INTERVAL_DATA_FORMAT, VERSION, DECIMAL_POINT,
 DATETIME_FORMAT, INTERVAL_CUT+)>
<!ELEMENT INTERVAL_DATA_FORMAT (#PCDATA)>
<!ELEMENT VERSION (#PCDATA)>
<!ELEMENT DECIMAL_POINT (#PCDATA)>
<!ELEMENT DATETIME_FORMAT (#PCDATA)>
<!ELEMENT CUT (RECORDER, CHANNEL, STARTTIME, STOPTIME, DST_PARTICIPANT,
 METER_START_READING, METER_STOP_READING, METER_MULTIPLIER,
 METER_OFFSET, PULSE_MULTIPLIER, PULSE_OFFSET, SPI, UOM, TIMEZONE,
 TIMESTAMP, ORIGIN?, DC_FLOW?, POPULATION?, WEIGHT?,
 TIME_ZONE_STANDARD_NAME?, DESCRIPTOR?, ACCEPT_REJECT_STATUS?,
 TRANSLATION_DATE?, INTERVAL)>
<!ELEMENT RECORDER (#PCDATA)>
<!ELEMENT CHANNEL (#PCDATA)>
<!ELEMENT STARTTIME (#PCDATA)>
<!ELEMENT STOPTIME (#PCDATA)>
<!ELEMENT DST_PARTICIPANT (#PCDATA)>
<!ELEMENT METER_START_READING (#PCDATA)>
<!ELEMENT METER_STOP_READING (#PCDATA)>
<!ELEMENT METER_MULTIPLIER (#PCDATA)>
<!ELEMENT METER_OFFSET (#PCDATA)>
<!ELEMENT PULSE_MULTIPLIER (#PCDATA)>
<!ELEMENT PULSE_OFFSET (#PCDATA)>
<!ELEMENT SPI (#PCDATA)>
<!ELEMENT UOM (#PCDATA)>
<!ELEMENT DC_FLOW (#PCDATA)>
<!ELEMENT TIMEZONE (#PCDATA)>
<!ELEMENT POPULATION (#PCDATA)>
<!ELEMENT WEIGHT (#PCDATA)>
<!ELEMENT TIME_ZONE_STANDARD_NAME (#PCDATA)>
<!ELEMENT DESCRIPTOR (#PCDATA)>
<!ELEMENT TIMESTAMP (#PCDATA)>
<!ELEMENT ORIGIN (#PCDATA)>
<!ELEMENT ACCEPT_REJECT_STATUS (#PCDATA)>
<!ELEMENT TRANSLATION_DATE (#PCDATA)>
<!ELEMENT INTERVAL (RECORDING+)>
<!ELEMENT RECORDING (VALUE, STATUS?, STARTTIME?)>
<!ELEMENT VALUE (#PCDATA)>
<!ELEMENT STATUS (#PCDATA)>
<!ELEMENT STARTTIME (#PCDATA)>
]>
```

## XML Compact Format

The XML “compact” format is an XML format used with interval data that has a smaller footprint than the XML standard format. The compact format is the same format used by the LSINTD Interval Data component, and can be used with the Oracle Utilities Adapter. See the *Oracle Utilities Energy Information Platform User's Guide* and *Oracle Utilities Energy Information Platform Installation and Configuration Guide* for more information about the *Oracle Utilities* Adapter.

### XML Compact File Format Examples

The following is an example of a standard interval data cut defined in the XML compact format.

```
<CUTS>
 <CUTEX>
 <HEADER
 RECORDER="1700"
 CHANNEL="1"
 STARTTIME="2003-02-24 00:00:00"
 STOPTIME="2003-02-24 23:59:59"
 START_READ=15000
 STOP_READ=17500
 M_MULT=1
 M_OFFSET=0
 SPI="1800"
 UOM="1"
 DESC="Pre-validation"]
 DCFLOW="D"
 ARS="00"
 TZSN="EST"
 ORIGIN="M"
 DST="Y"
 CHNSTATUS="0" />
 <INTS>
 <I V=12.00 S=" " E=" " />
 <I V=13.00 S=" " E=" " />
 <I V=14.00 S=" " E=" " />
 ...
 </INTS>
</CUTEX>
</CUTS>
```

Below is an example the compact XML format user to import data into an enhanced/generic interval data table that includes the CALCGROUP, MINIMUM, and MAXIMUM custom columns.

```
<CUTS>
 <CUTEX>
 <HEADER TIMESERIES="TEST,1" SPI="900" UOM="1" DST="Y" STARTTIME="2006-12-12
00:01:00" STOPTIME="2007-01-10 23:59:59" TDATE="1969-12-31 19:00:00" ORIGIN="C"
DCFLOW="" ARS="" START_READ="0.0" STOP_READ="0.0" POP="0.0" WEIGHT="0.0"
M_OFFSET="0.0" M_MULT="0.0">
 <CUSTOM CALCGROUP="1" MAXIMUM="10" MINIMUM="-10" />
 </HEADER>
 <INTS>
 <I V="16" S="7" />
 <I V="18" S="7" />
 <I V="16" S="7" />
 <I V="18" S="7" />
 <I V="15" S="7" />
 <I V="18" S="7" />
 <I V="16" S="7" />

 <I V="21" S="7" />
 <I V="20" S="7" />
 <I V="17" S="7" />
```



```

 </INTS>
 </CUTEX>
</CUTS>

```

Below is a similar example that includes a category.

```

<CUTS>
 <CUTEX>
 <HEADER TIMESERIES="PERCOT,1" CATEGORY="RAW" UOMCODE="01" STARTTIME="2005-
12-03T00:00:00" STOPTIME="2005-12-03T23:59:59" SPI="3600" DST="Y" P_MULT="1.0"
P_OFFSET="1.0">
 <CUSTOM CALCGROUP="1" MAXIMUM="1" MINIMUM="1"/>
 </HEADER>
 <INTS>
 <I V="2850" S=" " />
 <I V="3150" S=" " />
 <I V="2850" S=" " />
 <I V="3150" S=" " />
 <I V="2850" S=" " />
 <I V="3150" S=" " />
 <I V="2850" S=" " />
 <I V="3150" S=" " />
 <I V="2850" S=" " />
 <I V="3150" S=" " />
 <I V="2850" S=" " />
 <I V="3150" S=" " />
 <I V="2850" S=" " />
 <I V="3150" S=" " />
 <I V="2850" S=" " />
 <I V="3150" S=" " />
 </INTS>
 </CUTEX>
</CUTS>

```

## XML Compact File Format Element and Attribute Descriptions

The elements and attributes of the XML compact format are described below. All attributes correspond to elements described under **Standard XML Format** on page D-2. Defaults are inserted when the processing program encounters an omitted attribute.

**CUTS:** Optional element containing one or more CUTEX elements.

**CUTEX:** Element containing a single interval data cut.

Elements:

**HEADER:** Element containing header information for the interval data cut contained in the parent CUT element.

**Attributes:**

- **RECORDER** (key attribute): Recorder ID for the interval data cut. See **RECORDER (key element)** on page D-4 for detailed information.
- **CHANNEL** (key attribute): Channel number for the interval data cut. See **CHANNEL (key element)** on page D-4 for detailed information.
- **TIMESERIES** (key attribute): Identity of the parent record of the interval data. Used in place of the RECORDER and CHANNEL elements when defining interval data stored in enhanced interval data tables.
- **CATEGORY:** Optional category of the interval data. Used only with interval data stored in enhanced interval data tables..
- **STARTTIME** (key attribute): Start time for the interval data cut. Must be in ISO datetime format. See **STARTTIME (key element)** on page D-4 for detailed information.
- **STOPTIME:** Stop time for the interval data cut. Must be in ISO datetime format. See **STOPTIME** on page D-5 for detailed information.

- **TIMESTAMP:** (optional) Timestamp of the interval data cut. Must be in ISO datetime format. See **TIMESTAMP** on page D-10 for detailed information.
- **START\_READ:** (optional) Meter start reading for the interval data cut. See **METER\_START\_READING** on page D-6 for detailed information.
- **STOP\_READ:** (optional) Meter stop reading for the interval data cut. See **METER\_STOP\_READING** on page D-6 for detailed information.
- **M\_MULT:** (optional) Meter multiplier for the interval data cut. See **METER\_MULTIPLIER** on page D-6 for detailed information.
- **M\_OFFSET:** (optional) Meter offset for the interval data cut. See **METER\_OFFSET** on page D-7 for detailed information.
- **POP:** (optional) Population for the interval data cut. See **POPULATION** on page D-9 for detailed information.
- **WEIGHT:** (optional) Weight for the interval data cut. See **WEIGHT** on page D-9 for detailed information.
- **SPI** (Seconds per Interval): SPI for the interval data cut. See **SPI (Seconds per Interval)** on page D-7 for detailed information.
- **UOM** (Unit of Measure): UOM for the interval data cut. See **UOM (Unit of Measure)** on page D-8 for detailed information.
- **DESC:** (optional) Descriptor for the interval data cut. See **DESCRIPTOR** on page D-10 for detailed information.
- **DCFLOW:** (optional) DC Flow for the interval data cut. See **DC\_FLOW** on page D-8 for detailed information.
- **ARS:** (optional) Accept-Reject-Status for the interval data cut. See **ACCEPT\_REJECT\_STATUS** on page D-11 for detailed information.
- **TDATE:** (optional) Translation Date for the interval data cut. See **TRANSLATION DATE** on page D-11 for detailed information.
- **TZSN:** (optional) Time Zone Standard Name for the interval data cut. See **TIME\_ZONE\_STANDARD\_NAME** on page D-10 for detailed information.
- **ORIGIN:** (optional) Origin for the interval data cut. See **ORIGIN** on page D-10 for detailed information.
- **DST:** (optional) DST participant flag for the interval data cut. See **DST\_PARTICIPANT** on page D-5 for detailed information.
- **CHNSTATUS:** (optional) Status code for the interval data cut.

**Elements:**

- **CUSTOM:** Used to define custom columns added to enhanced/generic interval data tables. The format of this element is:

```
<CUSTOM <fieldname>=<value>/>
```

where:

- **<fieldname>** is an attribute that is the name of the custom column
- **<value>** is the value to be imported into the custom column

The **<fieldname>** attribute can be repeated for multiple custom columns.

For example, the following **<CUSTOM>** element includes import values for the CALCGROUP, MAXIMUM, and MINIMUM custom columns:

```
<CUSTOM CALCGROUP="1" MAXIMUM="10" MINIMUM="-10" />
```

## Interval Values

The interval values data is enclosed within the <INTS> node. Each individual interval value is enclosed within the <I> node, and consists of the V (Value) and optional S (STATUS) attributes.

**INTS:** Element containing interval values.

Elements:

**I:** Element containing a single interval value. The I node will not have an S attribute if and only if the interval status is a space.

**Attributes:**

- **V** (Interval Value): the actual recorded value for the interval
- **S** (Status Code): The status code for the interval
- **E** (Extended Status Code): The extended status code for the interval



# Appendix E

---

## Oracle Utilities Comma Separated Values (CSV) Interval Data Format

This appendix provides a detailed description of the Oracle Utilities Comma Separated Values (CSV) Interval Data Format, supported in Oracle Utilities programs such as Oracle Utilities Load Analysis, Oracle Utilities Billing Component, Oracle Utilities Rate Management, and Data Manager. This includes:

- **CSV Format**
- **Format Details**
- **Sample CSV File**

## CSV Format

This format is designed to support:

- Mixed interval lengths among cuts
- Daylight Savings Time
- International date and value formats (via tab characters for separators); the specific formats of dates and decimal values are solely dependent on the operating system in use.
- The possibility of a future enhancement for non-fixed interval lengths within cuts (with the use of Time tags for each interval).

The times that are defined in the export format will be expressed in the format designated by the **Short Date** and **Short Time** settings on the operating system. The language indicator and the decimal separator will also be read from the operating system. In all likelihood, the recipient of the export file generated by the producing Oracle Utilities application will be sharing the same language environment, so date and decimal format translation would not be necessary. If format translation is necessary, the date format, decimal separator, and language indicator found in the document description record of the export file could be used to translate the date and decimal formats to the desired format representation. The format translation that occurs in this scenario becomes solely the responsibility of the user, unless a future specification overrides the responsibility explicitly.

The format consists of two sections of data. The first section describes the export file format version and the regional settings. The second section contains data that appears in the same exact sequence as data would appear in the .LSE format, except that the data is separated by a semi-colon.

## General Description

Records in the CSV format are defined as a series of tab/comma/delimiter separated values followed by a carriage return/line feed pair. Each record begins with a record tag field that identifies that record type.

### General Rules

#### Order

Within each section, proper ordering is mandatory. Records in the Header and Detail sections should be arranged in the order as described in **Format Details** on page E-13. Trailing empty or null fields in any record cannot be omitted.

#### Fields

All fields, regardless of whether they are required or optional, must be enclosed with double quotation marks. Use the empty string to represent empty fields (“”).

#### Import Capability

The CSV Format file (\*.CSV) differs from an Enhanced Oracle Utilities (\*.LSE) file in two main respects.

- First, it contains a document description record that provides a date format and a time format. This record also contains a decimal separator, and the operating system 3-character locale identifier. Data can be re-imported from this file by translating the export date/time and decimal layouts to the appropriate format.
- Second, the data is formatted according to the date/time and decimal format that is specified in the document description record (see below).

## Document Description Record

The document description record contains the CSV format version, and the regional setting information. This required single record must precede all others in the export file. All fields within this record are required. The version number used should be related in some fashion to the version of the specification that defines the export format (e.g. 2.2). This version number must be changed when a syntactical change in the export format occurs for any reason. The importable option can be used to signify that a standard Application Program Interface (API) can process the data in this file.

```
[00000000][Document Type Identifier][Field Delimiter][Export Version][Field
Delimiter][Decimal Point Character][Field Delimiter][Language Indicator][Field
Delimiter][Date/Time Format][Field Delimiter][CSV Version][Record Delimiter]
```

### Notes:

1. The [Field Delimiter] is defined to be the character immediately following the first sequence that consists of a double quotation mark, followed by eight consecutive zero characters and then another double quotation mark (“00000000”). Characters preceding the first instance of this sequence are considered to be superfluous, and should be ignored. The [Field Delimiter] is most often a comma, but can be any delimiter character, or even non-printing characters such as the Tab character.
2. The [Document Type Identifier] identifies the format of the file.
3. The [Export Version] references the version of this document that the export was generated from. No assumptions about forward and/or backward compatibility can be assumed when comparing export files generated at differing versions. Likewise, CSV files with identical versions are said to be structurally compatible.
4. The [Decimal Point Character] cannot be the same character as the [Field Delimiter]. Should a conflict occur in the generation of the CSV file, the Decimal Separator takes precedence over the [Field Delimiter]. So, in case of a conflict, the [Field Delimiter] would be forced to change to an alternate character. Non-printable characters such as the tab character can be used in the place of commas, but please be aware that delimiters that differ from commas and semicolons may not be imported to some spreadsheets easily.
5. The [Language Indicator] is the unique 3-character abbreviation for a locale as defined by the operating system.
6. The [Date/Time Format] is expressed as a string consisting of a sequence of date and time elements strung together to form a date/time format (for example, mm/dd/yyyy hh:mm:ss). Predefined date/time format strings such as “Short Date” and “Short Time” are not legal formatting strings for the export format, because these formats are always relative to the operating system settings of the machine that generated the export, and therefore could not be determined from another computer.
7. The carriage return/line feed pair is always the [Record Delimiter]. Neither the [Field Delimiter], nor the [Decimal Separator] can be either a carriage return or a line-feed character.

## Field Relationships and Requirements

Relationships between fields exist in the CSV format. These are described along with detailed field requirements in this section. Defaults are inserted when the processing program encounters an omitted field (indicated by double commas) or white space between commas.

### Sort Code

#### Relationships:

This field has no relationship with any other field.



**Requirements:**

- Sort codes must contain eight digits, including leading 0s. Sort codes are restricted to integers between 00000000 and 99999999. However, 00000005 through 09999999 are not used, and may be ignored (see below).

**Note:** Sort code 00000000 can be used **only** in the Document Description record.

- Leading white space before the sort codes is not allowed.
- The first three header records are required. The fourth header record is optional.
- The document description record starts with sort code 00000000.
- Header records will have sort code 00000001 through 00000004.
- Column 1 of each header record must contain a 0.
- Records with Sort codes 00000005 through 05000000 are reserved. Records with Sort codes 05000001 through 09999999 may be present, but will be ignored.
- Data records must start with sort code 10000000, and progress sequentially (i.e. 10000001, 10000002...) through 99999999, or as high as needed to accommodate the data, with the maximum of 99999999.
- Column 1 of the data records must be greater than 0.

**Recorder/Customer Identifier (key element)****Relationships:**

None.

**Requirements:**

- Only letters, numbers, underscores, and hyphens will be accepted in this field; all other characters and embedded blanks will cause the record to be rejected.

**Note:** Lowercase letters will be translated to uppercase before they are stored in the database.

- Must be at least one character, and no more than 64 characters in length.

**Default:**

None; field is required.

**Channel (key element)****Relationships:**

None.

**Requirements:**

- Field values must be non-negative integer only.
- Maximum value is 32767.

**Default:**

None; field is required.

**Start Time (key element)****Relationships:**

- Must be a valid date/time value less than the value of the Stop Time.

**Requirements:**

- The date format has no limitations: it is defined in the **Document Description Record Format** on page E-13.
- The date itself must correspond to acceptable date ranges of the product in use (Data Manager, Oracle Utilities Billing Component, and other non-Oracle Utilities Load Analysis applications, as well as Oracle Utilities Load Analysis).
- No spaces, colons, or any ASCII value allowed between time components.

**Default:**

None; field is required.

**Stop Time****Relationships:**

- Must be a valid date/time value greater than the value of the Start Time.
- Value must agree with the number of interval values past the start time \* SPL.
- One second is subtracted if the value is equal to an even interval start time.

**Requirements:**

See Start Time.

**Default:**

None; field is required.

**DST Participant Flag****Relationships:**

None.

**Requirements:**

Valid values are Y, N, or A, indicating whether or not this record will be processed using DST adjustments.

Flag Value	Definition
Y	(DST Participant) The intervals recorded participate in DST adjustments and the intervals during DST reflect the time change.
N	(Does not participate in DST) The intervals recorded do not make any DST time changes.
A	(DST participant) The intervals have been adjusted to 24-hour days. (At the April time change, there should be a value at the time change (a place holder) of 0 with status code "9". At the October time change, there should be a combined value.) These cuts will be converted to a "Y" or "N", and are not stored in the database as "A".

**Default:**

As specified in the INTDDEFAULTDST configuration parameter setting, the INTDCONFIG.XML file, or the CSDST value in Oracle Utilities Load Analysis.

## Invalid Record Flag

**Relationships:**

None.

**Requirements:**

Valid values are Y or N, indicating whether or not the input programs should validate the incoming record.

Flag Value	Definitions
Y	The incoming record contains unvalidated data. In Oracle Utilities Load Analysis, a value of Y can be used to run this record through the validation routines.
N	The incoming record contains valid data (mark as valid).

**Default:**

Y.

## Meter Start Reading

**Relationships:**

Must be supplied if any of the other Meter fields (Meter Stop Reading, Meter Multiplier, or Meter Offset) are supplied.

**Requirements:**

If supplied, value must be a non-negative numeric value.

**Default:**

0.

## Meter Stop Reading

**Relationships:**

Must be supplied if any of the other Meter Fields (Meter Start Reading, Meter Multiplier, or Meter Offset) are supplied.

**Requirements:**

If supplied, value must be a non-negative numeric value.

**Default:**

0 (not supplied).

## Meter Multiplier

### Relationships:

Must be supplied if any of the other Meter fields (Meter Start Reading, Meter Stop Reading, or Meter Offset) are supplied.

### Requirements:

If supplied, value must be positive numeric. However, if no value is specified, 0 will be stored. Zero (0) is not a valid value for the meter multiplier, but its presence in the database indicates that no meter multiplier was supplied.

### Default:

None, 0 stored in the database if not supplied.

**Note:** If 0 is stored and/or observed in the database and a cut is exported to the LSE format, all meter fields for that cut will be absent. A zero stored in the database is indicative of “no meter information supplied”.

## Meter Offset

### Relationships:

Must be supplied if any of the other Meter fields (Meter Start Reading, Meter Stop Reading, or Meter Multiplier) are supplied.

### Requirements:

If supplied, value must be numeric.

### Default:

0 (not supplied).

## Pulse Multiplier

### Relationships:

None.

### Requirements:

Value must be positive numeric. Also, must be supplied if Pulse Offset (see below) is supplied.

### Default:

None, 0 stored in the database if not supplied.

**Note:** If 0 is stored and/or observed in the database and a cut is exported to the LSE format, all pulse fields for that cut will be absent. A zero stored in the database is indicative of “no pulse information supplied”.

## Pulse Offset

### Relationships:

If supplied, you must also supply Pulse Multiplier.

### Requirements:

Value must be numeric.

### Default:

None.

---

## Seconds Per Interval (SPI)

**Description:**

This field will store the interval duration if appropriate. An Intervals Per Hour (IPH) of 1 can be translated to a Seconds Per Interval of 3600.

**Relationships:**

Value of 0 indicates non-uniform recording duration and a full array of interval start values (one per recording) located in the data section. (This feature will be supported in a future release and is **not currently available**.)

**Requirements:**

- Value must be a positive integer.
- Valid values for SPI are 86400, 3600, 1800, 900, 300, and 60.

**Default:**

None; field is required.

**Unit of Measure**

The Oracle Utilities products will accept Units of Measure (UOMs) that are one to three digits in length. The **Units of Measure** on page B-16 are the ones supplied by Oracle.

**Description:**

See **Units of Measure** on page B-16.

**Relationships:**

None.

**Requirements:**

- Value should be a valid Unit of Measure, or a user-defined one. If the value entered is unknown to, 16 (Miscellaneous) will be stored.
- Value must be non-negative numeric.
- Maximum value is 32767.

**Default:**

None; field is required.

**Basic Unit Code****Description:**

This field is reserved for future use.

**Relationships:**

None.

**Requirements:**

Omit this field.

**Default:**

0.

## Times Zones West of GMT

**Description:**

This value tells programs accessing this record which time zone the values have been recorded in. Its value will be the difference in time, heading west, between Greenwich Mean Time and the represented time zone as measured in half hours. Eastern Standard Time should have a value of 10; Pacific = 16. A Value of -1 will indicate "Time Zone not available."

**Relationships:**

None.

**Requirements:**

Value must be -1 or a non-negative numeric between 0 and 47, that represents the time zones West of GMT measured in half-hour increments. Any record containing a time zone greater than 47 will be rejected.

**Default:**

-1.

## Population

**Description:**

For Oracle Utilities Load Analysis Statistical records only.

**Relationships:**

None.

**Requirements:**

Value may be any non-negative integer value.

**Default:**

0.

## Weight

**Description:**

For Oracle Utilities Load Analysis Statistical records only.

**Relationships:**

None.

**Requirements:**

Value may be any non-negative numeric value.

**Default:**

0.0.

## Time Zone Standard Name

**Description:**

Defines the Time Zone to which the cut is associated. The value must be one of EST, CST, MST, PST, or be defined in the LSCALENDAR.XML configuration file. Each value in this field must contain printable ASCII characters no longer than 32 bytes.

Only letters, numbers, underscores, and hyphens will be accepted in this field, all other characters and embedded blanks will cause the record to be rejected.

---

**Note:** Lowercase letters will be translated to uppercase before being stored

**Relationships:**

Must map exactly to one of EST, CST, MST, PST or an entry in the LSCALENDAR.XML file. Oracle Utilities Load Analysis will import without checking this file.

**Requirements**

Length: <= 32 bytes

**Descriptor****Description:**

The user may enter any descriptive information in this field. Any information entered into this field will be stored in the database verbatim as supplied.

**Relationships:**

None.

**Requirements:**

- Value must be 80 characters or fewer, and may contain commas.
- May start with blanks.
- If more than 80 characters are entered, truncation will occur and the first 80 will be used.

**Timestamp****Relationships:**

None.

**Requirements:**

- Format should be YYYYMMDDHHMMSSmmm. The milliseconds (mmm) may be omitted, in which case mmm will be set to 000.
- No spaces, colons, or any ASCII value allowed between time components.

**Default:**

Current time that data is loaded into database.

**Note:** It is strongly recommended that you leave this field empty and allow the system to input the TimeStamp for you.

**Origin****Relationships:**

None.

**Requirements:**

Must be one of: M (metered), P (profiled), C (computed), or S (Oracle Utilities Load Analysis Statistic).

**Default:**

M (metered).

## Interval Value

**Description:**

This field contains the actual recorded value for a time period.

**Relationships:**

Must have one status code entry per interval data value.

**Requirements:**

Any numeric value. (Up to 15 significant digits can be stored.)

**Default:**

0 (see Status Code, Relationships, below).

## Status Code

**Description:**

This field contains the status code for the preceding Interval Value.

**Relationships:**

Must have an entry for the Interval Value.

**Requirements:**

Any character will be accepted.

**Default:**

- This field defaults to a blank ( ' ') when omitted if the interval value is present.
- This field defaults to '9' when the Interval Value field is omitted.

## Interval Start

**Description:**

- This field indicates the starting time of the recording or the time when the recording took place.
- Format must be YYYYMMDDHHMMSS. (This feature will be supported in a future release and is **not currently available**.)

**Relationships:**

- Until this feature is supported, the SPI may not be omitted (represented by „).
- There must be one entry per Interval Value.

**Requirements:**

See Start Time, Requirements.

**Default:**

None.



## Format Details

### Document Description Record Format

ELEMENT	DESCRIPTION	MAXIMUM LENGTH IN BYTES (if applicable)	COMMENT
1	Sort Code	8	Must be 00000000 (Note: Character that immediately follows is the Field Delimiter.)
2	Document Type ID	1	LODESTAR Interval Data CSV format
3	Version	N/A	2.2
4	Decimal Separator	1	Usually "." or ","
5	Locale Abbreviation	5	Defined by OS
6	Date/Time Format	N/A	Date/Time format string

### First Header Record Format

ELEMENT	DESCRIPTION	MAXIMUM LENGTH IN BYTES (if applicable)	COMMENT
1	Sort Code	8	Must be 00000001
2	Recorder/ Customer Identifier	64	Letters, numbers, hyphens or underscores are acceptable values
3	Channel	5	Max is 32767
4	Start Time	N/A	Cut start date and time
5	Stop Time	N/A	Cut stop date and time
6	DST Participant Flag	1	Y/N/A
7	Invalid Record Flag	1	Y/N

## Second Header Record Format

ELEMENT	DESCRIPTION	MAXIMUM LENGTH IN BYTES (if applicable)	COMMENT
1	Sort Code	8	Must be 00000002 Default: None; field required
2	Meter Start Reading	N/A	Non-Negative Numeric Max: 9999999999999999.9999 Default: 0
3	Meter Stop Reading	N/A	Non-Negative Numeric Max: 9999999999999999.9999 Default: 0
4	Meter Multiplier	N/A	Positive Numeric (Optional) Max: 9999999999999999.9999 Default: 0
5	Meter Offset	N/A	Numeric (Optional) Max: 9999999999999999.9999 Default: 0
6	Pulse Multiplier	N/A	Positive Numeric (Optional) Max: 9999999999999999.9999 Default: 0
7	Pulse Offset	N/A	Numeric (Optional) Max: 9999999999999999.9999 Default: None; field required
8	Seconds Per Interval (SPI)	N/A	Positive Numeric Default: None; field required
9	Unit of Measure	N/A	Numeric Max: 32767 Default: None; field required
10	Basic Unit Code	N/A	Positive Numeric (Optional) Max: 9999 Default: 0
11	Time Zones West of GMT	N/A	Numeric. (Optional) Default: -1 Min: -1 Max: 47
12	Population	N/A	Positive Numeric (Optional) Max: 9999999999999999.9999 Default: 0.0
13	Weight	N/A	Positive Numeric (Optional) Max: 9999999999999999.9999 Default: 0.0
14	Time Zone Standard Name		Character (32) (Optional) If present, must be one of "EST", "CST", "MST", "PST", or definition must exist in LSCalendar.xml file.

### Third Header Record Format

ELEMENT	DESCRIPTION	MAXIMUM LENGTH IN BYTES (if applicable)	COMMENT
1	Sort Code	8	Must be 00000003
2	Descriptor	80	

### Fourth Header Record Format

ELEMENT	DESCRIPTION	MAXIMUM LENGTH IN BYTES (if applicable)	COMMENT
1	Sort Code	8	Must be 00000004
2	Timestamp	17	YYYYMMDDHHMMSSMM (optional) Default: current date
3	Origin	1	C, M, P, or S (optional)

### Data Record Format

ELEMENT	DESCRIPTION	MAXIMUM LENGTH IN BYTES (if applicable)	COMMENT
1	Sort Code	8	10000000 through 99999999
2	Interval Value		Interval value in locale-format
3	Status Code	1	
4	Interval Start	N/A	See Note 2
5	Interval Stop	N/A	See Note 2

### Notes

1. A data line may contain as many full data-status-time groups as desired, providing the record does not exceed a character count (record length) of 1024.
2. When this file is generated, the interval start time is always required. The interval stop time is required only for partial intervals.
3. Data spanning time must be contiguous and “missings” (or gaps) must be represented by value 0 and status '9'. For fixed-length SPIs, all intervals must be of the same duration, including “missings”. Only single intervals can appear in a single record.

## Sample CSV File

A sample \*.CSV file appears below:

```
"00000000","LODESTAR Interval Data CSV Format","2.2",".", "USE", "mm/dd/yyyy
hh:mm:ss"
"00000001","N1732","17","01/01/2000 00:00:00","01/31/2000 23:59:59","Y","Y"
"00000002","12.34","1234.56","1.0","100","1.0","0","900","04","","-1","",""
"00000003","","ACCEPTANCE TEST DATA NUMBER 1"
"00000004","","M"
"10000000","31","Q","01/01/2000 00:00:00",""
"10000001","32","Q","01/01/2000 00:15:00",""
"10000002","33","Q","01/01/2000 00:30:00",""
"10000003","34","Q","01/01/2000 00:45:00",""
"10000004","32","Q","01/01/2000 01:00:00",""
"10000005","30","Q","01/01/2000 01:15:00",""
```

## Differences with the .LSE format

There are a number of significant differences between the \*.LSE format and the \*.CSV format. These differences include:

- **Field Delimiters**
- **Field Types**
- **Document Description Record**
- **Date, Time, and Decimal Value Representation in Records**
- **Interval Data Records**

### Field Delimiters

Field delimiter in the CSV format is defined to be the character that immediately follows the first sort code ("00000000") in a file. They can be either printable or non-printable. The LSE format always uses a comma (",").

### Field Types

All fields in the CSV format must be enclosed in double quotation marks, regardless of the apparent data type. The LSE format doesn't require this.

### Document Description Record

The first record in a CSV file is always the document description record which always has a sort code of "00000000". This record provides date/time and decimal formats and locale information, as well as a version number so incompatibility can be determined. The LSE format doesn't include this record.

### Date, Time, and Decimal Value Representation in Records

All date, time, and decimal data in the CSV format uses the date/time and decimal format as defined in the document description record. The LSE file uses a specific decimal and time format.

### Interval Data Records

The interval start time is required for CSV files, but not for the \*.LSE format. Also, only a single interval value can be expressed in a record in CSV files. The interval data record contains an optional interval stop field that can be used to designate a partial interval.

## Compatibility Issues

All export files, including CSV files, have export format versions. Export files with different version numbers are inherently incompatible. A mapping algorithm is required to convert export files from one version to another. Any APIs created to read these CSV files must consult the export format version as basis for reading the files correctly. APIs should be flexible enough to react accordingly to older versions and use the proper mapping mechanism. A parser built to read a specific CSV format does not imply that it can automatically read an earlier version.



---

# Index

## Numerics

1 2-52

## A

Account class

Commercial 3-4

Residential 3-4

Account Component permissions 7-6, 7-7

Account Status records 3-2

Aggregating intervals

Aggregation methods per UOM C-16

attributes

transaction identifier 5-22

Average aggregate method 3-3, 3-5

## B

Bill Determinant Lookup codes 3-2

Bill History records 3-2

Bill History Value records 3-2

Bill History Value Table 3-2

BILLHISTORY Column 3-2

## C

CIS Account IDs 3-9

CIS Account Table 3-9

CISFORMAT.TXT 9-3

Code

in Jurisdiction Lookup code record 3-3

Control File 9-2, 9-3

Currency Presentation 1-6

Customer Component permissions 7-6

## D

Data Sources tab and functions

enabling 7-2

Database schema diagram A-1

Database sort order D-2

Date Presentation 1-6

Deletion of data sources

enabling 7-2

Deletion of positions

enabling 7-2

Deletion of roles

enabling 7-2

Deletion of sessions

enabling 7-2

Deletion of users

enabling 7-2

Directory Code field 3-10

DST adjustments D-5

## E

EMail field 3-10

End Use Lookup codes 3-3

Enhanced Direct Input File C-15

## F

Factor Table 3-11

Factor Value record 3-11

Factor value records 3-11

Fax field 3-10

Forecast Management Overview 1-1

## G

Group Category Lookup codes 3-12

## H

How to

Modify the .tls file 2-52

## I

ID field 3-9

Identifier 3-2

Insertion of accounts

enabling 7-6

INTDIMP

Inputs 8-13

International

Database Support 1-9

File Formats 1-9

International Support 1-5

Interrogations Lookup codes 3-12

Interval Data Format D-1

## J

Jurisdiction Code field 3-9, 3-11

## K

- kvarh 3-5
- KWH 3-5
- KWH-IN 2-21, 3-5
- KWH-OUT 2-21, 3-5

## L

- LAGGING KVARH 3-5
- Languages 1-5
- LEADING KVARH 3-5
- List Management menu item and function
  - enabling 7-8
- List Management Permission 7-8
- Locales 1-5
- Locales function
  - enabling 7-4
- Locales Permission 7-4
- LODESTAR.CFG 9-2
- Lookups
  - Account Status 3-2
  - Bill Determinant 3-2
  - Jurisdiction 3-2
  - LSCurrency 3-2
  - Operating Company 3-2
  - Revenue Code 3-2
  - SIC 3-2
  - UOM (Unit of Measure) 3-2
- LSCurrency table 1-6
- LTMH Monitor 6-77
- LTMH Server 6-76

## M

- Max aggregate method 3-3, 3-5
- Messaging Tool Tab 6-78
- Meter nodes D-6
- Meter Value records 3-2
- Multiple Currency Support 1-5, 1-8
  - Oracle Utilities Receivables Component 1-8
  - Oracle Utilities Receivables Component Reports 1-8
  - Other Oracle Utilities Products 1-8

## N

- Name
  - in Jurisdiction Lookup code record 3-3
- Name of File field 3-10
- Node Descriptions D-2
- Null value 3-9
- Numeric Presentation 1-6

## O

- Operating Company
  - definition 3-4
- Operating Company Code field 3-9, 3-11
- Oracle Utilities UOM codes 3-5
- Output File 9-2
- Override Code field 3-15
- Override Lookup codes 3-15

## P

- Pager field 3-10
- Permissions
  - Security 7-2
- Phone #1 field 3-10
- Phone #2 field 3-10
- Postwindow field 3-12
- Preview field 3-12
- Print Detail 9-9
- Prorate charges? 3-11

## R

- Relationships between nodes D-4
- Report Admin menu option
  - enabling 7-9
- Revenue Code Lookup codes 3-4
- RUNRS
  - Inputs 8-41
- Runtime Queue Tab 6-78
- Runtime Services Tab 6-77

## S

- Search Account menu item and function
  - enabling 7-5
- Search Customer menu item and function
  - enabling 7-5
- Search Meters menu item and function
  - enabling 7-6
- Search Recorders menu item and function
  - enabling 7-5
- Sessions tab and functions
  - enabling 7-2
- Standard Direct Input File C-15
- Standard Industrial Classification (SIC) Lookup codes 3-5
- Starting the LTMH Server 6-76
- Stop Billing
  - Automatic Only 5-13, 5-20
  - Manual Only 5-13, 5-21
- Stope Billing
  - for both Automatic and Approval Required modes 5-13, 5-21

## T

- tail attributes 5-22
- Title field 3-10
- Total aggregate method 3-3, 3-5
- Transaction File 9-2, 9-8
- Transaction File format
  - creating 9-1

## U

- Unit of Measure 3-2
- Unit of Measure (UOM) Lookup code 3-5
- Unit of Measure field 3-11
- UOM 3-2, 3-11
- User Context function
  - enabling 7-4
- User Context Permission 7-4
- Users tab and functions
  - enabling 7-2



## **X**

XML Interval Data Format D-1

